UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS

Novel Approaches to Cooperative Coevolution of Heterogeneous Multiagent Systems

Doutoramento em Informática
Especialidade Engenharia Informática

Jorge Miguel Carvalho Gomes

Tese orientada por:
Prof. Doutor Anders Lyhne Christensen
Doutor Pedro Lopes da Silva Mariano
Prof. Doutor Luís Miguel Parreira e Correia

Documento especialmente elaborado para a obtenção do grau de doutor

2017

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS

Novel Approaches to Cooperative Coevolution of Heterogeneous Multiagent Systems

Doutoramento em Informática
Especialidade Engenharia Informática

Jorge Miguel Carvalho Gomes

Tese orientada por:
Prof. Doutor Anders Lyhne Christensen
Doutor Pedro Lopes da Silva Mariano
Prof. Doutor Luís Miguel Parreira e Correia

Júri:
Presidente:
- Doutor Nuno Fuentecilla Maia Ferreira Neves

Vogais:
- Doutor Ágoston Endre Eiben
- Doutor Ernesto Jorge Fernandes Costa
- Doutor Anders Lyhne Christensen (orientador)
- Doutor João Paulo Marques da Silva
- Doutor Carlos Eduardo Ramos dos Santos Lourenço

Documento especialmente elaborado para a obtenção do grau de doutor

Instituição Financiadora: Fundação para a Ciência e Tecnologia (FCT) (SFRH/BD/89095/2012)

2017

Novel Approaches to Cooperative Coevolution of Heterogeneous Multiagent Systems

"As for natural evolution, consider the kangaroo. For thousands of years, people have been drawing odd animals, like chimeras and dragons. However, these were most often combinations and/or exaggerations of existing animals: strange – yes, original – no. In the meanwhile, before the discovery of Australia no one had imagined an animal with a pouch. Thus, the kangaroo is a metaphor for the truly original designs evolution can come up with."

A. E. Eiben
Grand Challenges for Evolutionary Robotics

Abstract

Heterogeneous multirobot systems are characterised by the morphological and/or behavioural heterogeneity of their constituent robots. These systems have a number of advantages over the more common homogeneous multirobot systems: they can leverage specialisation for increased efficiency, and they can solve tasks that are beyond the reach of any single type of robot, by combining the capabilities of different robots.

Manually designing control for heterogeneous systems is a challenging endeavour, since the desired system behaviour has to be decomposed into behavioural rules for the individual robots, in such a way that the team as a whole cooperates and takes advantage of specialisation. Evolutionary robotics is a promising alternative that can be used to automate the synthesis of controllers for multirobot systems, but so far, research in the field has been mostly focused on homogeneous systems, such as swarm robotics systems. Cooperative coevolutionary algorithms (CCEAs) are a type of evolutionary algorithm that facilitate the evolution of control for heterogeneous systems, by working over a decomposition of the problem. In a typical CCEA application, each agent evolves in a separate population, with the evaluation of each agent depending on the cooperation with agents from the other coevolving populations. A CCEA is thus capable of projecting the large search space into multiple smaller, and more manageable, search spaces. Unfortunately, the use of cooperative coevolutionary algorithms is associated with a number of challenges. Previous works have shown that CCEAs are not necessarily attracted to the global optimum, but often converge to mediocre stable states; they can be inefficient when applied to large teams; and they have not yet been demonstrated in real robotic systems, nor in morphologically heterogeneous multirobot systems.

In this thesis, we propose novel methods for overcoming the fundamental challenges in cooperative coevolutionary algorithms mentioned above, and study them in multirobot domains: we propose *novelty-driven cooperative coevolution*, in which premature convergence is avoided by encouraging behavioural novelty; and we propose *Hyb-CCEA*, an extension of CCEAs that places the team heterogeneity under evolutionary control, significantly improving its scalability with respect to the team size. These two approaches have in common that they take into account the exploration of the *behaviour space* by the evolutionary process. Besides relying on the fitness function for the evaluation of the candidate solutions, the evolutionary process analyses the behaviour of the evolving agents to improve the effectiveness of the evolutionary search.

The ultimate goal of our research is to achieve general methods that can effectively synthesise controllers for heterogeneous multirobot systems, and therefore help to realise the full potential of this type of systems. To this end, we demonstrate the proposed approaches in a variety of multirobot domains used in previous works, and we study the application of CCEAs to new robotics domains, including a morphological heterogeneous system and a real robotic system.

Keywords: Evolutionary robotics; cooperative coevolutionary algorithms; heterogeneous multirobot systems

Resumo

Sistemas multi-agente distribuídos, e sistemas multi-robô em particular, são sistemas em que vários agentes autónomos (robôs) cooperam para atingir um objectivo comum. Os sistemas multi-robô têm como grande vantagem conseguir atingir um nível de robustez, eficácia, e competência, que vai para além das capacidades de qualquer robô único. O potencial destes sistemas já foi demonstrado experimentalmente numa série de aplicações, incluindo busca e salvamento, construção, agricultura, vigilância, exploração e mapeamento, etc.

Um tipo de sistemas multi-robô que ainda está na sua infância, apesar do seu enorme potencial, são os sistemas heterogéneos, em que os robôs constituintes têm diferentes morfologias e/ou controladores. A heterogeneidade pode ser vantajosa para tarefas robóticas que possam ser naturalmente decompostas em sub-tarefas, e pode ser aproveitada para resolver tarefas que estejam fora do alcance de qualquer tipo único de robô. Em muitas situações, é preferível ter vários robôs simples com capacidades complementares, em vez de ter um (ou vários) robôs com uma elevada complexidade. A heterogeneidade pode, de facto, ser essencial para alcançar sistemas com as capacidades necessárias para realizar tarefas mais ambiciosas, e capazes de lidar com a complexidade do mundo real.

A maioria da investigação em sistemas multi-robô foca-se, no entanto, em sistemas homogéneos, em que todos os robôs têm a mesma morfologia e o mesmo controlador. Um dos grandes desafios em conceber sistemas heterogéneos é a síntese do controlador para cada tipo de robô, de modo a que haja uma adequada divisão de tarefas, e que o grupo coopere e tire partido das capacidades conjuntas. Programar manualmente controladores para sistemas multi-robô distribuídos é, de forma geral, desafiante, pois o comportamento que se pretende do grupo tem que ser destilado em regras de controlo para cada um dos robôs. Este desafio é exacerbado em sistemas heterogéneos, pois os graus de liberdade aumentam em função do número de agentes no sistema.

O uso de algoritmos evolutivos representa uma alternativa promissora para conceber de forma automática controladores para sistemas multi-robô, tendo já sido aplicada para solucionar uma grande variedade de tarefas robóticas. Contudo, mais uma vez, a grande maioria da investigação tem-se focado em sistemas homogéneos. Evoluir controladores para sistemas homogéneos é inerentemente mais simples, pois apenas é necessário evoluir um controlador, que é depois copiado para todos os robôs do grupo. A evolução de controladores para sistemas heterogéneos implica a evolução de vários controladores em simultâneo, de modo a que funcionem bem uns com os outros, o que aumenta significativamente o espaço de procura.

Os algoritmos de co-evolução cooperativa (CCEAs) constituem um tipo de algoritmos evolutivos especialmente adequados para lidar com grandes espaços de procura, pois operam sobre uma decomposição do problema. Numa aplicação típica de um CCEA, os controladores dos diferentes agentes co-evoluem simultaneamente em populações separadas. Os agentes em evolução em cada população são recompensados em função de quão bem eles funcionam com os agentes das outras populações, direcionando assim o processo evolutivo para a evolução de equipas de sucesso. Os algoritmos de co-evolução cooperativa têm, no entanto, algumas limitações conhecidas:

Convergência prematura para estados de equilíbrio: Num CCEA, as avaliações dos indivíduos são relativas, uma vez que o resultado da sua avaliação é contextualmente dependente do estado das outras populações em co-evolução. O mesmo indivíduo tanto pode ser considerado bom como mau, dependendo dos outros indivíduos com o qual ele é avaliado. Devido a esta relatividade, os CCEAs tendem a convergir prematuramente para estados de equilíbrio que não correspondem necessariamente a soluções óptimas.

Escalabilidade com o tamanho da equipa: Tipicamente cada agente co-evolui numa população separada, pois é a decomposição natural do problema. Esta abordagem, no entanto, não escala muito bem para grandes equipas: a complexidade computacional aumenta linearmente com o número de populações; as diferentes populações podem evoluir comportamentos muito semelhantes, o que é redundante; e podem surgir dificuldades em discernir o impacto de um só agente no desempenho de toda a equipa.

Demonstração limitada em sistemas multi-robô: Os CCEAs têm sido principalmente estudados em problemas de optimização numérica e teoria de jogos. Também já foram aplicados numa variedade de domínios multi-robô, mas focando sempre o mesmo tipo de sistema: morfologicamente homogéneos, e em ambiente simulado.

O objectivo deste trabalho é avançar na direcção de métodos que possam ser utilizados para sintetizar controladores para sistemas multi-robô heterogéneos, com a complexidade necessária para lidar com tarefas do mundo real. Neste sentido, focamo-nos em desenvolver novos métodos para mitigar as limitações dos algoritmos co-evolutivos. O nosso trabalho não se foca em solucionar nenhuma tarefa robótica em particular. Procuramos, por outro lado, desenvolver algoritmos que possam ser facilmente aplicados a uma grande variedade de tarefas multi-robô cooperativas, e procuramos obter resultados que sejam indicativos do desempenho geral dos algoritmos propostos. Respondemos a três principais questões neste trabalho:

É possível evitar a convergência prematura nos algoritmos de co-evolução cooperativa através da procura de novidade?

Conseguir evitar a convergência prematura é um passo essencial para que os CCEAs possam ser aplicados a tarefas mais complexas. Uma abordagem que se tem revelado valiosa para evitar a convergência prematura em algoritmos não-co-evolutivos é a procura de novidade. Neste tipo de abordagem, o processo evolutivo é guiado pela procura constante de novas soluções: os indivíduos são recompensados por exibir comportamentos diferentes daqueles que já foram encontrados até ao momento, em vez de serem apenas recompensados de acordo com a função objectivo. Isto resulta num processo evolutivo divergente, que tipicamente evita a convergência para um máximo local.

Nesta tese, propomos e estudamos pela primeira vez CCEAs baseados em procura da novidade. As abordagens propostas são comparadas com o CCEA tradicional baseado numa função objectivo, e com outras técnicas populares usadas para evitar convergência prematura em CCEAs. Os resultados obtidos mostraram consistentemente que a melhor forma de implementar a procura de novidade em CCEAs consiste em obter a novidade do indivíduo com base na novidade da equipa com o qual ele foi avaliado (NS-Team). O valor de novidade é adicionalmente combinado com o tradicional valor da função objectivo, usando um processo de selecção

multi-objectivo. A abordagem NS-Team consegue evitar a convergência prematura porque evita estados de equilíbrio – a métrica de novidade promove a evolução de indivíduos que geram novos comportamentos de equipa, contrariando assim a pressão de as populações se acomodarem umas às outras. Além da capacidade de evitar a convergência prematura, a abordagem proposta é também capaz de descobrir uma diversidade valiosa de soluções num único processo evolutivo. A generalidade desta abordagem é suportada pelos bons resultados obtidos em diversos domínios multi-robô.

É possivel usar algoritmos de co-evolução cooperativa para evoluir controladores para sistemas multi-robô reais, e para sistemas morfologicamente heterogéneos?

De modo a melhor compreender o potencial e limitações dos algoritmos co-evolutivos, é necessário avaliá-los em novos domínios. Em primeiro lugar, demonstramos pela primeira vez o uso de CCEAs para sintetizar controladores para um sistema real. O estudo baseia-se numa tarefa cooperativa de perseguição, num sistema multi-robô aquático de superfície que já tinha sido utilizado anteriormente em outros estudos de robótica evolutiva. Os controladores foram co-evoluídos em simulação, e depois transferidos e sistematicamente avaliados no sistema real. De modo geral, o desempenho das equipas em simulação correspondeu ao desempenho das equipas quando avaliadas no sistema real, mostrando que os CCEAs podem ser aplicados com sucesso a sistemas robóticos reais. O uso da co-evolução baseada na procura da novidade revelou-se também eficaz, evoluindo uma diversidade de soluções que o CCEA tradicional nunca conseguiu descobrir.

Em segundo lugar, estudamos pela primeira vez o uso de CCEAs para desenvolver controladores para um sistema morfologicamente heterogéneo, composto por um robô aéreo e um terrestre. Os resultados obtidos mostraram que, quando os robôs conseguem facilmente cooperar um com o outro, os CCEAs conseguem evoluir boas soluções. No entanto, nas tarefas em que é mais difícil estabelecer cooperação, os CCEAs convergiram frequentemente para solução não-cooperativas com baixo desempenho. A procura da novidade ajudou a mitigar o problema da convergência prematura, mas não conseguiu ser totalmente eficaz nos casos mais extremos. Este estudo mostra a importância do desenvolvimento mútuo das capacidades dos robôs em co-evolução, e a importância de facilitar a cooperação entre eles. Quando estas condições são satisfeitas, o processo co-evolutivo é capaz de lidar com heterogeneidade arbitrária nas populações.

É possível melhorar a escalabilidade dos algoritmos de co-evolução cooperativa permitindo heterogeneidade dinâmica nas equipas?

Para que seja viável usar CCEAs para evoluir controladores para grandes equipas, é necessário melhorar a escalabilidade destes algoritmos. Colocar o nível de heterogeneidade sob controlo do processo de aprendizagem é uma abordagem promissora para lidar com este problema. Ao permitir sub-equipas homogéneas dentro da equipa, é possível reduzir o número de controladores que têm que ser sintetizados. Neste trabalho, propomos o Hyb-CCEA, o primeiro algoritmo co-evolutivo a colocar a composição da equipa sob evolução, abandonando a correspondência um-para-um entre agentes e populações. No algoritmo proposto, cada população pode ser atribuída a vários agentes, criando assim uma sub-equipa homogênea. Durante o processo evolutivo, as populações que estão a evoluir agentes com comportamentos

semelhantes são fundidas, diminuindo assim a heterogeneidade da equipa. Complementarmente, as populações podem também ser divididas de forma estocástica para aumentar a heterogeneidade, garantindo assim a exploração de diferentes composições de equipa ao longo do processo evolutivo.

Os resultados mostram que o Hyb-CCEA é capaz de encontrar composições de equipa adequadas para a tarefa em questão, desde equipas totalmente homogéneas a totalmente heterogéneas. O algoritmo proposto é capaz de diminuir significativamente o número de populações no processo co-evolutivo, mitigando o problema da evolução redundante (diferentes populações a evoluir os mesmos comportamentos). Estas vantagens resultam numa grande redução no número de recursos necessários para alcançar soluções, quando comparado com um CCEA tradicional, completamente heterogéneo. O Hyb-CCEA conseguiu ainda alcançar frequentemente soluções com um nível de qualidade nunca alcançado pelo CCEA tradicional.

Acima de tudo, o trabalho apresentado nesta tese confirma o potencial dos algoritmos de co-evolução cooperativa como uma ferramenta promissora para evoluir controladores para sistemas multi-robô heterogéneos. Os métodos propostos nesta tese estão alinhados com a nova corrente de investigação que defende que as técnicas de robótica evolutiva devem ir para além da simples optimização numérica. Através da análise dos comportamentos que estão a ser produzidos no processo evolutivo, conseguimos evitar a convergência prematura (co-evolução baseada na procura da novidade) e melhorar a escalabilidade relativamente ao número de agentes (Hyb-CCEA). Este trabalho aproxima-nos de métodos para sintetizar de forma eficaz controladores para sistemas multi-robô heterogéneos. A capacidade de desenvolver controladores para este tipo de sistemas, que é actualmente um desafio tanto para técnicas automáticas como manuais, é um passo essencial para a adopção da heterogeneidade em sistemas multi-robô, e para concretizar o potencial deste tipo de sistemas.

Palavras-chave: Robótica evolutiva; algoritmos de co-evolução cooperativa; sistemas multi-robô heterogéneos

Acknowledgements

I would like to express my gratitude to my advisors, Anders Christensen and Pedro Mariano, who guided me in this journey. Anders Christensen has been my advisor since the Master's thesis, and has shaped me as an independent researcher. He taught me how to think critically and how to write science, he challenged my ideas, he thoroughly revised dozens of papers, and he was always there when needed. For all this, I am deeply grateful to him. Pedro Mariano offered different perspectives on my work, which were very helpful and helped me think. I would also like to thank Luís Correia, who joined the advisory team at the very end and allowed me to conclude the PhD, for accepting to join in such circumstances.

I am also grateful to the groups and research institutes that hosted me – BioISI, Agents and Systems Modeling group (former LabMAg), at FCUL, and Instituto de Telecomunicações, BioMachines Lab, at ISCTE-IUL. They provided me with a place to stay, with the necessary resources, and computational infrastructure which was essential to this work. But most importantly, they gave me the chance to meet great people in this journey. A sincere thank you to my fellow research colleagues and friends Miguel Duarte, Fernando Silva, Nuno Henriques, Sancho Oliveira, Davide Nunes, Vasco Costa, and others I've had the pleasure to meet.

This work would not have been possible without the financial assistance from a number of institutions, to which I express my gratitude. To Fundação para a Ciência e Tecnologia (FCT), the agency that directly funded me for four years, with the PhD grant SFRH/BD/89095/2012. To Instituto de Telecomunicações for funding me during the beginning of this PhD (grant PEst-OE/EEI/LA0008/2011) and for supporting scientific missions to several conferences (grant UID/EEA/50008/2013). To BioISI, Biosystems and Integrative Sciences Institute (UID/Multi/04046/2013), for also supporting me in scientific missions. To the CORATAM project (FCT grant EXPL/EEI-AUT/0329/2013), which supported the realisation of some of the experiments described in this thesis, and also supported me in scientific missions.

Finally, I must thank my family for their support, and my parents in particular. They trusted me and always supported me in whatever path I chose, in any way I needed. I am only at this point today because they gave me the conditions and education to achieve so. Last but not least, I thank my life partner and my best friend, Viviana. For enduring my doubts, my absence at times, and for always being there for me. For everything.

To all the people who trusted me, invested in me, and helped me grow.

Jorge Gomes
March 2017

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
BC	Behaviour Characterisation
CCEA	Cooperative Coevolutionary Algorithms
EA	Evolutionary Algorithm
EGT	Evolutionary Game Theory
ER	Evolutionary Robotics
MAS	Multiagent System
MOEA	Multi-Objective Evolutionary Algorithm
MRS	Multirobot System
NS	Novelty Search
QD	Quality Diversity

Chapter 1

Introduction

1.1 Motivation

An intelligent agent is an autonomous entity which observes through sensors, and acts upon an environment using actuators, directing its activity towards achieving goals (Russell and Norvig, 1995). Autonomous robots are intelligent agents with a physical body, which can autonomously sense and act upon the real physical world. In the last decade, autonomous robots have began to be used in real-world applications, moving from fiction to reality. Autonomous robots are currently used to clean the floor of millions of homes worldwide (iRobot Roomba[1], Jones, 2006), to gather data in the middle of oceans (Liquid Robotics Wave Glider[2], Daniel et al., 2011), to track and film people from the air (DJI Phantom 4[3]), and they have just began to appear in the public roads as self-driving cars (e.g. Tesla Autopilot[4]). All these state-of-the-art systems, as well as the vast majority of autonomous robots, operate as single-robot systems, in which a single robot performs the task alone, interacting only with the human operator. The next step for autonomous robots, which is gathering considerable attention[5], is to move to systems composed of multiple cooperating autonomous robots.

Distributed multiagent systems, and multirobot systems in particular, are systems where multiple autonomous agents (robots) cooperate to achieve a given goal. These systems take inspiration from natural societies (Şahin, 2005; Nitschke, 2005b), as they try to leverage the principles of self-organisation, division of labour, and cooperation, to achieve distributed autonomous systems where the whole is greater than the sum of its parts. Distributed multirobot systems have the potential to achieve a level of robustness, efficiency through parallelism, and combined competences that go beyond the capabilities of any single robot (Parker, 2008).

Multirobot systems have shown their potential in a number of domains, including search and rescue (Sheh et al., 2016), construction (Werfel et al., 2014), security and surveillance (Guo et al., 2004), agriculture (Pitla, 2012), and mapping and exploration of unknown environments (Howard et al., 2006). Few real-world implementations of these multirobot systems have, however, actually occurred (Parker, 2008). Multirobot systems are relatively recent, and their realisation faces a number of technological challenges. A significant amount of research is therefore still needed

[1] http://www.irobot.com/

[2] https://www.liquid-robotics.com/

[3] https://www.dji.com/phantom-4

[4] https://www.tesla.com/autopilot

[5] Technical Committee on Multi-Robot Systems of the IEEE Robotics and Automation Society, founded in 2014: http://multirobotsystems.org/

to bring these systems closer to real-world application. The expectation is that multirobot systems will find their way into practical and cost-effective applications as the supporting technologies continue to mature (Parker, 2008).

One type of multirobot systems that is still in its infancy, despite its vast potential, is heterogeneous systems. Heterogeneous multirobot systems are a type of multirobot systems characterised by the morphological and/or behavioural heterogeneity of their constituent robots. It has been argued that heterogeneity might be fundamental to achieve more complex systems capable of dealing with realistic and more ambitious tasks (Dorigo et al., 2013; Parker, 2008). Indeed, the advantages of heterogeneity can be confirmed by looking at many natural systems and social organisations, where behavioural heterogeneity is leveraged to increase efficiency and promote self-organisation through the division of labour and specialisation (Simpson, 2011). Canonical examples include eusocial insects (Wilson and Hölldobler, 2005), such as ants, bees, or wasps, where thousands of individuals cooperate under a clear division of tasks; small teams of mammals, such as lions, that take highly specialised roles when hunting preys (Anderson and Franks, 2001); and even human organisations, in which division of labour has long been a key concept in the improvement of work efficiency (Smith, 1776).

Most of the multirobot studies conducted so far have, however, focused on *homogeneous* multirobot systems (Bayındır, 2016; Parker, 2008), in which all the robots of the team share the same morphology, and typically also the same controller. The potential of heterogeneity in multirobot systems has not yet been thoroughly explored, which is pertinent given the significant advantages of heterogeneity as demonstrated in natural societies. Previous studies have shown that morphological and/or behavioural heterogeneity can be advantageous for robotics tasks that can be naturally decomposed into a set of complementary sub-tasks (Balch, 1998; Nitschke, 2008). Heterogeneity can additionally be leveraged to accomplish tasks that are beyond the reach of any single type of robot, either due to morphological or behavioural limitations (Kengyel et al., 2015). In many situations, it might be advantageous to have multiple simple robots with complementary capabilities, rather than monolithic robots with a high complexity (Grabowski et al., 2000).

One challenge in designing heterogeneous systems is the synthesis of control for each type of robot, so that the team as a whole cooperates and takes advantage of the total set of capabilities (Parker, 1998; Parker, 2008). In fact, designing control for distributed multirobot systems in general has proven to be a challenging endeavour. Manually designing the control for the individual units of a group requires the decomposition of the system-level behaviour into behaviour rules for the individual robots (Brambilla et al., 2013). This decomposition is typically not trivial, as it is hard to know beforehand which individual rules will result in the desired self-organised group behaviour. The design problem is exacerbated when dealing with heterogeneous systems, as the degrees of freedom increase as more different agents are present in the system. This relative difficulty of conceiving control for heterogeneous systems might partially explain why the field of multirobot systems has traditionally been dominated by homogeneous systems.

Evolutionary algorithms represent an effective way of automatically designing control for multirobot systems (Trianni et al., 2008), as the candidate solutions are evaluated according to their group-level behaviour, thus avoiding the decomposition design problem. Evolutionary robotics (Nolfi and Floreano, 2000) have been successfully used to solve a wide range of robotics tasks, from single-robot tasks to swarm robotics tasks with hundreds of agents. But once again, the vast majority of the evolutionary robotics studies have focused on homogeneous systems. Evolving

control for homogeneous systems is inherently easier, as only a single controller has to be evolved, which is then copied to all the robots of the team. Evolving control for a heterogeneous team requires different controllers to be simultaneously evolved so that they work well with one another. This naturally causes the search space to grow proportionally to the number of different agents (Panait and Luke, 2005a), which can be detrimental to the performance of the evolutionary algorithm.

Cooperative coevolutionary algorithms (CCEAs) (Popovici et al., 2012; Potter and De Jong, 2000) have shown to be a promising approach to tackle the increased search space, since they operate over a decomposition of the problem. In CCEAs, the different agents are simultaneously coevolved in separate populations, effectively resulting in a solution composed of cooperating subcomponents. The individuals of each population are rewarded according to how well they cooperate with the individuals from the other populations, thus driving the evolutionary process towards the evolution of successful collaborations. Cooperative coevolutionary algorithms have a number of advantages over the non-coevolutionary approaches for the design of heterogeneous multiagent systems (Panait and Luke, 2005a), such as the capability of working over a decomposition of the problem (the evolution of a team) into more manageable sub-problems (the evolution of each agent), and facilitating the emergence of agent specialisations.

Cooperative coevolutionary algorithms are, however, also associated with a number of challenges, including:

Premature convergence: In a CCEA, the individual fitness evaluations are subject to a dynamic fitness landscape, given that the result of the evaluation is contextually dependent on the state of the other coevolving populations. This type of evaluation starkly contrasts with traditional evolutionary algorithms where the fitness landscape is static. Due to this variability in the individual fitness evaluations, it has been shown that CCEAs are especially prone to premature convergence to sub-optimal solutions (Panait, 2010).

Scalability with the team size: In a typical application of a CCEA, each agent coevolves in a separate population, as this corresponds to the natural decomposition of the problem. Such approach, can however, cause scalability issues when dealing with large teams. The computational complexity increases linearly with the number of populations (Potter and De Jong, 2000); different isolated populations might evolve very similar agent behaviours (redundant learning (D'Ambrosio et al., 2010)), which is a waste of resources; and credit assignment issues might arise (Agogino and Tumer, 2008), as the impact of a single agent in the performance of the whole team can become unperceivable.

Limited demonstration in multirobot systems: CCEAs have been mostly studied in test-bed problems such as function optimisation problems or abstract games based on game theory (Popovici et al., 2012). CCEAs have also been applied to a wide array of multirobot domains, but focusing always the same type of system: morphologically homogeneous, and on a simulated environment (with more or less abstraction).

In this thesis, we focus on developing novel methods for overcoming these fundamental challenges in cooperative coevolutionary algorithms, and study them in the domain of multirobot systems: we propose novelty-driven cooperative coevolution, which attempts to overcome premature convergence by encouraging behavioural novelty; and we propose *Hyb-CCEA*, an extension of CCEAs that put the

team heterogeneity under evolutionary control, significantly improving its scalability with respect to the team size. The two proposed approaches have in common that they take into account the exploration of the *behaviour space* by the evolutionary process. Besides relying on the fitness function for the evaluation of the candidate solutions, the evolutionary process relies on the analysis of the agents' behaviour to improve its effectiveness. This concept of behaviour space exploration has been gaining increasing traction in evolutionary robotics. The field has started to move beyond purely black-box optimisation (Doncieux and Mouret, 2014; Doncieux et al., 2015; Silva et al., 2016a), with a large number of studies showing the importance of promoting behavioural exploration and novelty.

We propose cooperative coevolutionary algorithms that adopt this paradigm of behaviour exploration, and study how it can be used to effectively mitigate coevolutionary-specific issues. The ultimate goal of our research is to achieve methods that can be more effectively used to synthesise controllers for heterogeneous multirobot systems, thus helping to realise the full potential of this type of systems. To this end, we demonstrate the proposed approaches in a variety of multirobot domains used in previous works, and we study the application of CCEAs to new robotics domains, including a real robotic system and a morphologically heterogeneous system.

1.2 Problem Statement

The goal of our research is to develop and study cooperative coevolutionary algorithms, in the direction of methods that can be used to synthesise controllers for heterogeneous multirobot systems in complex real-world tasks. To this end, we work towards circumventing key issues in this class of algorithms, and demonstrating them in new robotics domains. We focus on three main research questions in this thesis, stated below.

Research Question 1

How can behavioural novelty be leveraged to avoid premature convergence in cooperative coevolutionary algorithms?

CCEAs have been shown to be attracted to stable states instead of near-optimal solutions (Panait, 2010), due to the evolutionary dynamics between the coevolving populations. Premature convergence to stable states should be distinguished from the typical local convergence problems that plague non-coevolutionary algorithms (Panait et al., 2006b), as it is a consequence of the interplay between the coevolving populations, not necessarily a deceptive or rugged fitness landscape. This issue can compromise the use of CCEAs as optimisation tools for challenging multiagent tasks (Wiegand, 2003). Existing techniques for overcoming premature convergence have a very high computational cost, which makes them impractical in domains where the fitness evaluations are computationally expensive – such as multirobot systems. Improving convergence to near-optimal solutions, without significantly increasing computational complexity, is therefore an essential step towards CCEAs that can be effectively applied to problems of the complexity required to solve real-world tasks.

Recent works with non-coevolutionary algorithms have shown that a promising approach to avoid premature convergence is to drive the evolutionary process towards behavioural novelty, instead of a fixed fitness objective (Doncieux and Mouret, 2014; Silva et al., 2016b). This type of approaches, in which the novelty

search algorithm (Lehman and Stanley, 2011a) stands out, work by continuously encouraging divergence in the evolutionary process, thus promoting the exploration of the behaviour space and preventing convergence to a single optimum. In this thesis, we will study how novelty search can be implemented in cooperative coevolutionary algorithms, and whether it can be used to prevent premature convergence to stable states.

Research Question 2

Can cooperative coevolutionary algorithms be effectively used to evolve controllers for real multirobot systems, and morphologically heterogeneous systems?

The domain of application and demonstration of CCEAs has mainly been limited to function optimisation (Potter and De Jong, 1994), abstract games (Wiegand et al., 2002), and simulated morphologically homogeneous multirobot systems (Nitschke, 2008), with varying degrees of abstraction. To uncover the full potential and limitations of CCEAs, it is necessary to evaluate them in additional types of multirobot systems that are commonly found in the robotics state of the art (Parker, 2008). In this thesis, we study the application of CCEAs to the control of a real multirobot system, where the control is evolved in simulation and then transferred to a real system operating in realistic conditions. To the best of our knowledge, this is the first study of a real robotic system with controllers synthesised by a CCEA. We also study for the first time the challenges of using CCEAs to synthesise control for morphologically heterogeneous systems, a type of systems with considerable potential for real-world applications (Dorigo et al., 2013; Parker, 2008).

Research Question 3

Can the scalability of cooperative coevolutionary algorithms be improved through dynamic team heterogeneity?

CCEAs inherently have scalability issues with respect to the number of different agents in the system. Using the standard coevolutionary architecture, the computational complexity increases with the number of agents in the team (Potter and De Jong, 2000), and there is a significant amount of resources wasted in redundant learning (D'Ambrosio and Stanley, 2008). Improving the scalability with respect to the number of agents is fundamental so that it becomes viable to use CCEAs to evolve control for heterogeneous multiagent systems with a large number of agents.

Previous works have shown that dynamic team heterogeneity can be a powerful approach to improve the scalability of multiagent learning (Bongard, 2000; D'Ambrosio and Stanley, 2008; Hara, 1999). By allowing homogeneous sub-teams inside the team, the number of controllers that need to be produced decreases, thus increasing scalability. Such concept has, however, only been demonstrated with non-coevolutionary algorithms, where the controllers for the whole team are encoded in a single monolithic chromosome. In this thesis, we study how dynamic team heterogeneity can be implemented in a cooperative coevolutionary algorithm, and whether it can offer significant advantages with respect to scalability.

Our research does not focus on solving any specific robotics task. Instead, we strive to develop general CCEA algorithms that can be easily applied to a broad range of cooperative multirobot tasks, and we aim at obtaining results that are indicative of the algorithms' general performance. In agreement with this principle,

our work respects the fundamental motivation behind evolutionary robotics (Doncieux et al., 2015; Harvey et al., 1997): the proposed evolutionary algorithms should facilitate the evolution of effective solutions from scratch, introducing as few biases from the experimenter as possible. In the next section, we summarise how the aforementioned challenges were addressed in our research.

1.3 Contributions and Publications

Novelty-driven Cooperative Coevolution

In the first part of this thesis (Chapter 3), we study the problem of premature convergence to stable states in CCEAs. Following the lead on the recent successes of novelty-driven evolutionary techniques in non-coevolutionary algorithms, we propose and study the first CCEA algorithms driven by behavioural novelty. Our experiments show that novelty-driven cooperative coevolution is an effective technique for mitigating premature convergence to stable states, outperforming the traditional CCEA algorithms and the existing techniques for overcoming premature convergence. We also show that novelty-driven coevolution can yield a wide diversity of successful solutions, as opposed to the traditional fitness-driven CCEAs that converge to a single solution.

This work has resulted in publications on the leading conference on autonomous agents and multiagent systems (*AAMAS*), and in one of the highest impact journal in the field of evolutionary computation (*Evolutionary Computation* from MIT Press):

- J. Gomes, P. Mariano, and A. L. Christensen (2014a). "Avoiding Convergence in Cooperative Coevolution with Novelty Search". In: *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*. IFAAMAS, pp. 1149–1156

- J. Gomes, P. Mariano, and A. L. Christensen (2017b). "Novelty-driven Cooperative Coevolution". In: *Evolutionary Computation*. *In press*

Serving as support and background work for novelty-driven cooperative coevolution, we conducted a number of additional studies focused on novelty-driven algorithms. The results obtained in these studies are not specific to novelty-driven cooperative coevolution, and can be used in any algorithms driven by behavioural novelty. We devised generic behaviour characterisations that can be used in multi-robot tasks; we proposed an approach for systematically deriving behaviour characterisations based on a formal description of the task; and we conducted a comprehensive empirical study on the parameters and implementation choices of novelty search algorithms. These studies are not described in detail in this thesis for reasons of brevity and consistency, but a summary of these contributions is presented in Section 2.5. These studies have been published in top international conferences in the field of evolutionary computation:

- J. Gomes and A. L. Christensen (2013). "Generic Behaviour Similarity Measures for Evolutionary Swarm Robotics". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 199–206
  Nominated for Best Paper Award

- J. Gomes, P. Mariano, and A. L. Christensen (2014e). "Systematic Derivation of Behaviour Characterisations in Evolutionary Robotics". In: *International*

Conference on the Synthesis and Simulation of Living Systems (ALife). MIT Press, pp. 212–219

- J. Gomes, P. Mariano, and A. L. Christensen (2015c). "Devising Effective Novelty Search Algorithms: A Comprehensive Empirical Study". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 943–950

In an exploratory study, we also evaluated the applicability of novelty-driven evolution to *competitive* coevolutionary algorithms. These results are not reported in this thesis, since the focus is exclusively on *cooperative* coevolutionary algorithms. The study resulted in the following publication:

- J. Gomes, P. Mariano, and A. Christensen (2014c). "Novelty Search in Competitive Coevolution". In: *Parallel Problem Solving from Nature (PPSN)*. vol. 8672. LNCS. Springer, pp. 233–242

Study of CCEAs in Additional Multirobot Domains

In the second part of this thesis (Chapters 4 and 5), we demonstrate the potential of CCEAs in multirobot domains different of those commonly used in CCEA studies. We studied both traditional CCEA algorithms as well as novelty-driven cooperative coevolution, further validating the general applicability of the proposed approach. Our experiments focus on a real aquatic multirobot system, operating in real-world conditions, and on a simulated morphologically heterogeneous system composed of aerial and ground robots. These studies are the first application of CCEAs to evolve controllers for such types of systems, contrasting with the previous works that focused exclusively on simulated morphologically homogeneous systems. These experiments have been published in two international conferences in evolutionary computation, receiving a nomination for the best paper award in both cases, and in an international journal:

- J. Gomes, P. Mariano, and A. L. Christensen (2015a). "Cooperative Coevolution of Morphologically Heterogeneous Robots". In: *European Conference on Artificial Life (ECAL)*. MIT Press, pp. 312–319
  Nominated for Best Paper Award

- J. Gomes, P. Mariano, and A. L. Christensen (2016a). "Challenges in cooperative coevolution of physically heterogeneous robot teams". In: *Natural Computing*, pp. 1–18

- J. Gomes, M. Duarte, P. Mariano, and A. L. Christensen (2016b). "Cooperative Coevolution of Control for a Real Multirobot System". In: *Parallel Problem Solving from Nature (PPSN)*. Springer, pp. 591–601
  Nominated for Best Paper Award

Dynamic Team Heterogeneity

In the third part of this thesis (Chapter 6), we focus on the improvement of scalability of CCEAs, regarding the number of agents. We propose Hyb-CCEA, an approach that extends coevolutionary algorithms with operators that enable dynamic team heterogeneity. Hyb-CCEA puts the number of coevolving populations and the team composition under evolutionary control, using the behaviour similarity between the different agents in the team to regulate the team composition. Our experiments

show that Hyb-CCEA can significantly improve the effectiveness of the evolutionary process, converging to a team composition suitable for the given task, ranging from fully homogeneous to fully heterogeneous.

The Hyb-CCEA approach has been published in the leading conference in autonomous agents and multiagent systems (AAMAS), a short paper with the approach has been published in a workshop in the leading evolutionary computation conference (GECCO), and the work is currently under consideration for publication in the *IEEE Transactions of Evolutionary Computation* journal:

- J. Gomes, P. Mariano, and A. L. Christensen (2015b). "Cooperative Coevolution of Partially Heterogeneous Multiagent Systems". In: *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*. IFAAMAS, pp. 297–305
- J. Gomes, P. Mariano, and A. L. Christensen (2015d). "Hyb-CCEA: Cooperative Coevolution of Hybrid Teams". In: *Genetic and Evolutionary Computation Conference Companion (Evolving Collective Behaviors in Robotics Workshop)*. ACM Press, pp. 1251–1252
- J. Gomes, P. Mariano, and A. L. Christensen (2017a). "Dynamic Team Heterogeneity in Cooperative Coevolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation*. Under revision

The general theme and goals of this thesis have additionally been presented to the scientific community at two conferences, in a peer-reviewed workshop and a doctoral symposium:

- J. Gomes, P. Mariano, and A. L. Christensen (2014b). "Diversity-based Coevolution of Behaviourally Heterogeneous Multirobot Systems". In: *Workshop on Nature-inspired Techniques for Robotics at PPSN*
  Won Best Student Presentation
- J. Gomes (2014). "Evolution of heterogeneous multirobot systems through behavioural diversity". In: *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*. IFAAMAS, pp. 1729–1730

1.4 Other Contributions to Evolutionary Robotics

During the course of the Ph.D., a number of other studies have been conducted as a result of scientific collaborations. Although these works are not directly related to the objectives of this thesis, they represent contributions with a significant impact in the field of evolutionary robotics.

Evolution of Control for a Real Swarm of Aquatic Surface Robots

In a recent project (*CORATAM – Control of Aquatic Drones for Maritime Tasks*[6]), we have have demonstrated, for the first time, that evolutionary robotics can be successfully applied to evolve control for swarm robotics systems that operate in real-world, uncontrolled conditions (Figure 1.1). We used evolutionary techniques to synthesise control for an aquatic swarm robotics system, performing a variety of

[6]Funded by Fundação para a Ciência e Tecnologia (FCT, MCTES, Portugal), with the grant EXPL/EEI-AUT/0329/2013

FIGURE 1.1: The aquatic robotic swarm with 10 units, performing a homing task with evolved controllers. Image from (Duarte et al., 2016b).

canonical swarm robotics tasks, as well as more applied and complex tasks. This project resulted in several publications in international conferences, a journal publication, as well as an award for *Best Robot Video* in the prestigious AAAI Conference on Artificial Intelligence:

- M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, and A. L. Christensen (2016b). "Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots". In: *PLoS ONE* 11 (3), e0151834
- M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, and A. L. Christensen (2016e). "Unleashing the Potential of Evolutionary Swarm Robotics in the Real World". In: *Genetic and Evolutionary Computation Conference Companion (GECCO)*. ACM Press, pp. 159–160
- M. Duarte, J. Gomes, V. Costa, S. M. Oliveira, and A. L. Christensen (2016d). "Hybrid Control for a Real Swarm Robotics System in an Intruder Detection Task". In: *European Conference on the Applications of Evolutionary Computation (EvoApps)*. Springer, pp. 213–230
- M. Duarte, J. Gomes, V. Costa, T. Rodrigues, F. Silva, V. Lobo, M. Marques, S. M. Oliveira, and A. L. Christensen (2016a). "Application of Swarm Robotic Systems to Marine Environmental Monitoring". In: *IEEE/MTS OCEANS*. IEEE Press, pp. 1–8
- A. L. Christensen, M. Duarte, V. Costa, T. Rodrigues, J. Gomes, F. Silva, and S. M. Oliveira (2016). "A Sea of Robots". In: *AAAI Conference on Artificial Intelligence*. AAAI Press. URL: `https://www.youtube.com/watch?v=JBrkszUnms8`
  Peer-reviewed video, **winner of the Best Robot Video Award**

This project has additionally been covered by the media in several outlets with wide exposure, including:

- IEEE Spectrum (11-02-2016). *AAAI Video Highlights: Drones Navigating Forests and Robot Boat Swarms*. URL: `http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/aaai-video-highlights-drones-navigating-forests-and-robot-boat-swarms`

- Daily Mail Online (03-02-2016). *Will SWARMS of smart surveillance ships soon spy from the sea? Researchers reveal self learning ships that can think for themselves*. URL: http://www.dailymail.co.uk/sciencetech/article-3430481/Will-SWARMS-smart-surveillance-ships-soon-spy-sea-Researchers-reveal-self-learning-ships-think-themselves.html
- ZDNet (05-02-2016). *Drones of the sea learn to swarm*. URL: http://www.zdnet.com/article/drones-of-the-sea-learn-to-swarm-video/

Evolution of Control for Robots with Complex Locomotor Systems

In a separate line of research, we have focused on how to evolve control for robots with complex locomotor systems, such as legged robots (Figure 1.2). We proposed an approach (*EvoRBC*) in which we combine the evolution of locomotion repertoires with the evolution of high-level task-oriented controllers. Our study is amongst the first to be able to evolve task-oriented control for robots with a complex locomotor system. This research has, so far, resulted in a publication in the top conference in evolutionary computation, and is currently under revision for publication in the IEEE Transactions on Evolutionary Computation journal:

- M. Duarte, J. Gomes, S. M. Oliveira, and A. L. Christensen (2016c). "EvoRBC: Evolutionary Repertoire-based Control for Robots with Arbitrary Locomotion Complexity". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 93–100
- M. Duarte, J. Gomes, S. M. Oliveira, and A. L. Christensen (2017). "Evolution of Repertoire-based Control for Robots with Complex Locomotor Systems". In: *IEEE Transactions on Evolutionary Computation*. Under revision

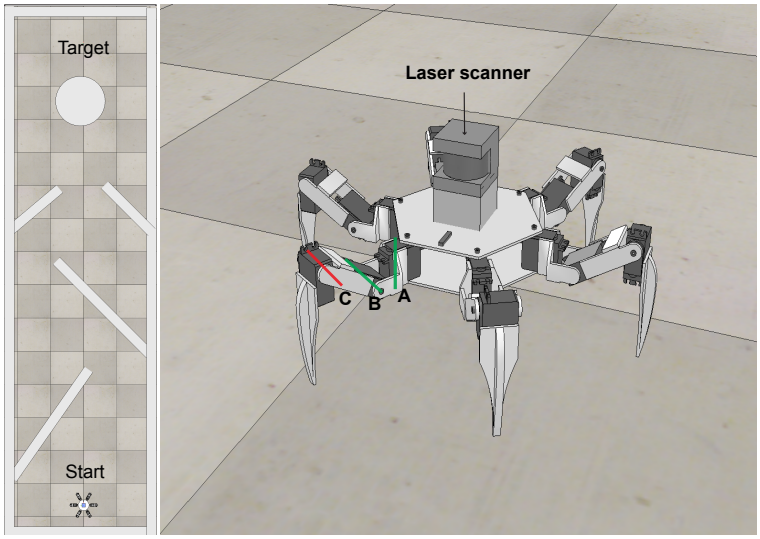


FIGURE 1.2: Maze navigation task (left) and the simulated hexapod (right) used in the validation experiments of EvoRBC. Image from (Duarte et al., 2017).

1.5 Thesis Structure

We begin in Chapter 2 by reviewing the current state of the art in the fields of heterogeneous multirobot systems; evolutionary robotics and their application to heterogeneous systems; cooperative coevolutionary algorithms; and novelty-driven evolutionary algorithms. We present the key concepts and definitions related to our work, and discuss the most recent and relevant advances in the field.

In Chapter 3, we propose and comprehensively study novelty-driven cooperative coevolution. We begin by describing the proposed algorithm and the analysis tools used in this study. We then thoroughly study the proposed approach in a classic multirobot task, and finally demonstrate the potential of the approach in two additional simulated multirobot tasks of greater complexity. In Chapter 4, we present a set of experiments in which novelty-driven cooperative coevolution, and CCEAs in general, are validated in a real multirobot system, operating in realistic conditions. We describe the methodology used to evolve the controllers in simulation, and analyse how the performance of the evolved teams was affected when transferred to the real robotic system. In Chapter 5, we study the application of CCEAs, and the proposed novelty-driven cooperative coevolution, to morphologically heterogeneous multirobot systems. We study the challenges specific to applying CCEAs to morphologically heterogeneous systems, and show how novelty-driven cooperative coevolution can help to mitigate these challenges, comparing it to other competing approaches. In Chapter 6, we propose the Hyb-CCEA approach, aimed at improving the scalability of coevolutionary algorithms regarding the number of agents. The proposed approach is extensively studied in an abstract domain, and then validated in two simulated multirobot domains.

Finally, in Chapter 7, we conclude by discussing the contributions of this thesis within the current state of the art, the limitations of our work, and we present directions for future work. Appendix A lists the experimental parameters and details for the experiments described in the other chapters, and in Appendix B, we briefly describe the software framework that was developed to support this work.

Chapter 2

Background

Distributed multirobot systems are inspired by the observation of societies, which are based on division of labour, cooperation and communication (Arai et al., 2002). If such collective organisation can benefit societies (both human and animal), multirobot systems could also benefit from those same concepts (Jones and Mataric, 2005). Multirobot systems have several advantages over single-robot systems (Cao et al., 1997), with the most common motivations for developing multirobot systems being (Parker, 2008):

- The task complexity is too high for a single robot to accomplish.
- The task is inherently distributed.
- Building several simple robots is much easier than constructing a single powerful robot.
- Multiple robots can solve problems faster due to their inherent parallelism.
- A high degree of robustness can be achieved through redundancy.

In this chapter, we introduce the main concepts approached in this thesis, and review the state of the art. We begin by discussing the advantages and uses of heterogeneous multirobot systems. We then present the field of evolutionary robotics, and how evolutionary algorithms have been used to automatically synthesise control for multirobot systems. We discuss the different approaches that can be used to evolve control for heterogeneous multiagent systems in particular, and how cooperative coevolutionary algorithms (CCEAs) stand as one of the most promising approaches. We define and describe CCEAs, discuss the challenges and limitations that have been found in previous studies, and briefly present the algorithmic extensions that have been proposed, as well as the previous domains of application. Finally, we introduce evolution driven by behaviour novelty and behaviour exploration, which are the basis for the methods proposed in this thesis.

2.1 Heterogeneous Multirobot Systems

The majority of the current studies on collective robotics focuses on homogeneous multirobot systems (Waibel et al., 2009), i.e., systems where all the agents in the group share the same controller and morphology. Swarm robotics systems (Bayındır, 2016; Brambilla et al., 2013; Şahin, 2005) are a popular class of multirobot systems that are characterised by decentralised control, use of local information, and self-organised global behaviour. In a typical swarm robotics system, see Figure 2.1 for examples, relatively simple units rely on self-organisation to display collectively intelligent behaviour. The behavioural differentiation between the robots

(a) Groß et al. 2006

(b) Rubenstein et al. 2012

(c) Duarte et al. 2016b

FIGURE 2.1: Examples of morphologically and behaviourally homogeneous robotic swarms.

in the system is typically not defined a-priori (i.e., the system is homogeneous), but can emerge as the result of the interactions among the robots and the environment (Bayındır, 2016; Brambilla et al., 2013; Ferrante et al., 2015). While such coordination mechanism can be a powerful form of self-organisation, it might be insufficient when the task at hand requires division of labour and the specialisation of agents in sub-tasks (Bernard et al., 2015; Montanier et al., 2016). Dorigo et al., (2013) argue that heterogeneity might be fundamental to achieve more complex systems capable of dealing with realistic tasks.

Researchers have recently begun focusing on heterogeneous multirobot systems, which are characterised by the morphological and/or behavioural diversity of their constituent robots. Behavioural heterogeneity is commonly employed to allow behaviour specialisation within the agent team (Campbell and Wu, 2011). In morphologically heterogeneous systems, on the other hand, robots have different actuation and sensing capabilities, and collaborate to take advantage of the collective set of capabilities (Dorigo et al., 2013).

2.1.1 Behaviourally Heterogeneous Systems

In behaviourally heterogeneous systems, robots can share the same morphology and capabilities, but have different controllers. The robots in the system can thus display different specialised behaviours. Allowing behavioural heterogeneity is, however, typically a trade-off — it allows a significant increase in the capabilities and efficiency of the group, but it comes at the price of an increased search space, which can complicate the design of the system. In the design of collective behaviour systems, it

remains an open research question as to which tasks are most appropriately solved using specialisation.

In an early work using learning agents, Balch, (1998) studied the emergence of specialisations in three multirobot domains (robot foraging, robot soccer, cooperative movement). It was shown that heterogeneity improved the performance and learning speed of the teams in certain tasks, but it was harmful when the task at hand was more based on parallelism than cooperation — i.e., when a single agent could reasonably perform the task alone. Similar results have been obtained in a number of other studies (Li et al., 2002; Murciano et al., 1997; Yong and Miikkulainen, 2009), which show that behavioural specialisation can be beneficial or not depending on the task and the environment. Overall, these studies suggest that if the task can be naturally decomposed into a set of complementary sub-tasks, then specialization is often beneficial for increasing collective task performance (Balch, 1998; Bernard et al., 2015; Nitschke, 2008). A counterargument is presented by Kengyel et al., (2015), who describe experiments where there are four behaviour types available, and an evolutionary algorithm optimises the team composition (how many agents of each type) for solving the given task. The evolved compositions are often nontrivial and even counterintuitive, but they outperform any purely homogeneous composition. These results reveal that the task decomposition might not always be clear beforehand, which can complicate the decision on whether a system should be behaviourally homogeneous or heterogeneous.

In the domain of multirobot systems, behaviourally heterogeneous systems have shown their potential in a number of tasks (Levi and Kernbach, 2010; Parker, 1994), such as robot soccer (Iocchi et al., 2003; Luke et al., 1998), collective surveillance (Colby and Tumer, 2015b), collective construction (Nitschke, 2012; Trueba et al., 2011), cooperative foraging (Bernard et al., 2015; Montanier et al., 2016; Nitschke et al., 2010), predator-prey pursuit (Nitschke et al., 2012b), movement in formation (Balch and Arkin, 1998), among others (Nitschke, 2008).

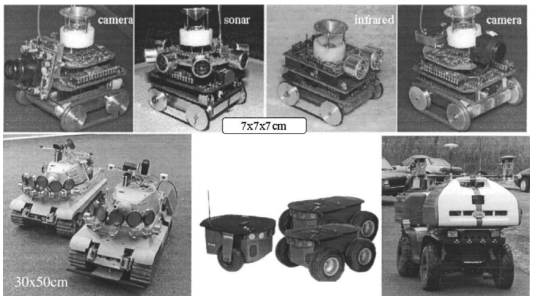
2.1.2 Morphologically Heterogeneous Systems

In morphologically heterogeneous systems, the agents in the system have different morphologies, which includes both systems where the robots have different locomotor systems and physical characteristics, as well as systems where the robots are physically similar but have different sensing and/or actuator capabilities (Parker, 2008). In either case, the cooperation between morphologically heterogeneous robots can enable the achievement of tasks that are beyond the reach of a single type of robot. Typically, morphologically heterogeneous systems are also genetically heterogeneous, as the robots need different controllers in order to cope with the specificities of their morphology.

The Swarmanoid project (Dorigo et al., 2013) studied morphologically heterogeneous robotic swarms. This project focused on swarms composed of robots with different morphologies to operate in three-dimensional human-centric environments. Three types of robots were built (see Figure 2.2a): (i) *eye-bots*, flying robots specialised in sensing and analysing the environment; (ii) *hand-bots*, with capabilities for climbing vertical surfaces of walls or objects; and (iii) *foot-bots*, specialised on moving through rough terrain and transporting objects. Some of the studies conducted within the Swarmanoid project include:

(a) Dorigo et al. 2013

(b) Chaimowicz et al. 2005



(c) Grabowski et al. 2000

(d) Howard et al. 2006

FIGURE 2.2: Examples of morphologically heterogeneous multirobot systems.

Mathews et al., (2010): A group of *foot-bots* cooperate with one *eye-bot* to overcome an obstacle in the environment. The *eye-bot* communicates with the *foot-bots*, providing the directions necessary for the *foot-bots* to assemble in the correct morphology and overcome the obstacle.

Ducatelle et al., (2011): A swarm of *foot-bots* and *eye-bots* accomplishes a task of indoor navigation, based on stigmergic interactions between the robots.

Dorigo et al., (2013): A swarm composed of robots of all types (*foot*, *hand* and *eye*) accomplishes a search and retrieval task in a complex 3-D environment. The swarm must first find the shelves containing relevant objects and then transport the objects from the shelves back to the deployment area.

Other works outside Swarmanoid have also shown the potential of cooperation between ground and aerial robots, especially in search-and-rescue tasks (Duan and Liu, 2010; Lacroix and Le Besnerais, 2011). Aerial robots have a privileged perspective of the environment, and can therefore be used to assist ground robots in a variety of tasks. Sukhatme et al., (2002), for instance, demonstrated a helicopter robot cooperating with two ground robots in tasks involving payload deployment and recovery, cooperative localization, and reconnaissance and surveillance tasks (Figure 2.2b). Chaimowicz et al., (2005) and Hsieh et al., (2007) demonstrated teams of aerial and ground robots cooperating for surveillance applications in urban environments.

Morphologically heterogeneous systems also encompass systems composed of robots of a similar nature (e.g., ground robots only). Heterogeneity can be used to reduce the cost of the system, by assigning different sensor/actuator capabilities to different robots, which can then cooperate to take advantage of each other's capabilities. Grabowski et al., (2000) showed such an approach in a mapping and exploration task, using multiple types of ground robots equipped with complementary sensors (Figure 2.2c). In (Parker et al., 2004), capable leader robots assist sensor-limited robots in navigating indoor environments. Howard et al., (2006) extended this approach to a task where few complex robots cooperate with a large number of inexpensive robots to map the environment and establish a sensor network (Figure 2.2d). In a different application domain, Simmons et al., (2001) demonstrated the use of heterogeneous robots for autonomous assembly and construction tasks relevant to space applications.

Another compelling reason to study morphological heterogeneity is that, in some cases, heterogeneity may be a necessity arising from practical constraints (Parker, 2008). It might be difficult to build a truly homogeneous robot team, since each copy of the same model of robot can vary widely in capabilities due to differences in sensor tuning, calibration, wear and tear, etc. In other practical scenarios, a multirobot system might have to be composed of the different types of robots that are currently available to perform the task (Blumenthal and Parker, 2004; Candea et al., 2001; Jones et al., 2006). In both these situations, behavioural control must take the differences in robot capabilities into account.

2.2 Evolutionary Robotics

Manually designing controllers for the individuals in a multirobot system is a challenging endeavour, since the desired system behaviour has to be decomposed into behavioural rules for the individual robots, taking into account the interactions among the system components (Panait and Luke, 2005a; Parker, 2008; Stone and Veloso, 2000). This requires discovering the relevant interactions between the individual robots and between them and the environment, which will ultimately lead to the emergence of global coordinated behaviour. Evolutionary robotics is often used to overcome this difficulty and automate the design process (Trianni, 2008).

Evolutionary robotics is a field of research that employs evolutionary computation to generate robots that adapt to their environment through a process analogous to natural evolution (Silva et al., 2016a). Evolutionary algorithms are optimisation methods that use operators for reproduction, mutation, and selection to artificially evolve solutions for a given problem. The individuals in a population play the role of candidate solutions to the problem, and a fitness function determines which individuals are best suited for solving the problem. The evolution of the population takes place through the repeated application of the genetic operators, see Figure 2.3. The seminal works of evolutionary robotics (Beer and Gallagher, 1992; Floreano and Mondada, 1994; Harvey et al., 1993) proposed an approach where the agent controller is based on a neural network, which is evolved through genetic algorithms. In these works, the authors argue that evolutionary robotics can be more adequate than traditional symbolic AI in the task of developing adaptive behaviours, since it promotes the shaping of the agents to their environment, and does not depend on the ability of the designer to consider all the possible contingencies. Using the evolutionary approach, the intelligent behaviours emerge from the interaction between an agent's internal control mechanisms and its external environment, rather than

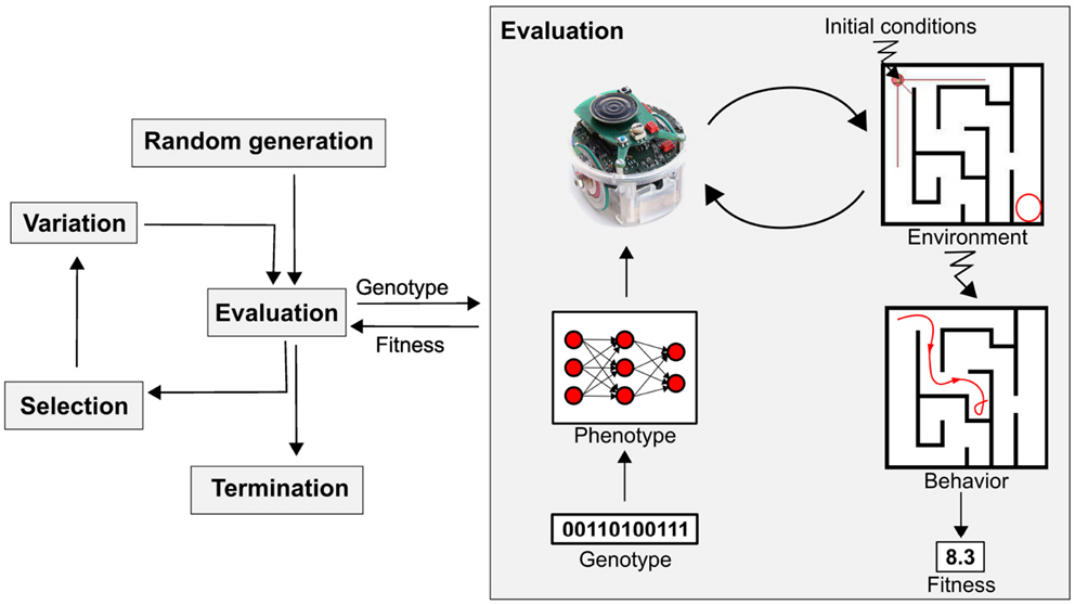


FIGURE 2.3: Principal workflow of evolutionary robotics. Image from (Doncieux et al., 2015).

from an agent's ability to reason explicitly with symbolic representations of states. The foundations of evolutionary robotics were later established by Nolfi and Floreano, (2000), who describe the basic concepts, methodologies, and a set of empirical experiments of varying complexity.

Evolutionary robotics can be a valuable approach for designing controllers for multirobot systems, because the application of evolutionary computation can eliminate the need for manual decomposition of the desired group behaviour (Francesca and Birattari, 2016; Trianni, 2008). The multirobot system is evaluated as a whole, and relies on the evolutionary process to synthesise the controller(s) that will be used locally by the individuals. The experimenter only has to provide the system-level fitness function. Evolutionary robotics has been applied to a large number of multirobot tasks, in both genetically homogeneous (Bayındır, 2016) and heterogeneous systems (Panait and Luke, 2005a; Waibel et al., 2009). While most studies are conducted exclusively in simulation or laboratory environments, the potential of evolutionary robotics has also been demonstrated in realistic environments (Duarte et al., 2016b).

The fitness evaluation of each candidate solution usually consists of running a simulation with the robots in their environment, using the controller(s) encoded in the chromosome, and measuring their performance in the task according to the user-provided fitness function (Nelson et al., 2009). In *offline* evolution, which is the most common approach, the evolutionary process is conducted offline, with the evaluation phase conducted in simulated environments. The chosen evolved controllers can then be transferred to the real robotic system (Jakobi, 1997; Miglino et al., 1995). In a parallel line of research – *online* (or *embodied*) evolution (Watson et al., 1999) – the entire evolutionary process is conducted directly in the robots, thus allowing continuous adaptation and avoiding potential issues of transferring controllers from simulation to reality (the *reality gap*) (Silva et al., 2016b). Online evolution, however, tends to be burdensome and take a long time, making it impractical for most tasks. In this thesis, we focus exclusively on *offline* evolution.

The controllers evolved for the robots are typically artificial neural networks

(ANNs) (Silva et al., 2016a), which process the sensory data and output the actuator values. ANNs are well-suited to control autonomous robots for a number of reasons, including (Floreano and Mondada, 1994): (i) tolerance to noise, making them good candidates for mediating between sensors and actuators with intrinsic noise; (ii) capability of approximating any function given the right network architecture, thus supporting complex input-output mappings; and (iii) they are well-suited for artificial evolution, since small changes in weights of the network typically correspond to small changes in the input-output mapping, thus allowing the evolutionary algorithms to progress gradually towards the solution.

A vast number of algorithms for neuroevolution have been proposed in previous studies (Floreano et al., 2008), varying the genetic operators, the chromosome encoding, the type of neural network evolved, among others. In the experiments described in this thesis, we use two different approaches for evolving neural networks:

Direct encoding of fixed topologies: The simplest approach to neuroevolution, where the neural network under evolution has a fixed topology, provided by the experimenter. The weights of the neural network are directly encoded as real values in the chromosome of the individual. A genetic algorithm evolves the chromosomes using standard operators such as Gaussian mutation, one-point crossover, and tournament selection.

NEAT: NeuroEvolution of Augmenting Topologies (NEAT) (Stanley and Miikkulainen, 2002) is a widely used neuroevolution algorithm, and one of the most successful approaches in the evolutionary robotics domain. NEAT simultaneously optimises the connection weights and evolves the topology of the neural network through incremental complexification. NEAT is therefore able to evolve networks with an arbitrary topology. It employs speciation and fitness sharing to maintain high genotypic diversity in the population, and to protect topological innovations.

2.3 Evolution of Heterogeneous Multiagent Systems

Heterogeneity in a multiagent system may significantly increase its capabilities, but this comes at the price of increased complexity (D'Ambrosio et al., 2010; Stone and Veloso, 2000). When optimisation techniques are used to design multiagent control, heterogeneity complicates the learning process (Dorigo et al., 2013; Panait and Luke, 2005a), as the size of the search space becomes proportional to the number of different agents. Behavioural control must integrate the abilities of different agents for them to work in synergy towards achieving a common goal. In many cases, however, information about which specialisations are needed to solve the given task, or if they are needed at all, is not available (Bongard, 2000). This means that it is typically not possible to evolve each agent independently from the others. To solve the control design problem, it is necessary to pursue a holistic approach, in which interactions between different agents are taken into account from the very beginning of the learning process (Dorigo et al., 2013; Panait and Luke, 2005a).

To evolve controllers for collective robotics systems, two main approaches can be considered (Panait and Luke, 2005a): *team learning* and *concurrent learning*.

Team learning

In *team learning* (also referred to as *team encoding* (Lichocki et al., 2013)), there is only a single learner involved, meaning that the whole team behaviour is encoded in a

single genome. This approach is typically used when evolving homogeneous systems, where all the agents use a copy of the same controller, but can also be used in heterogeneous systems, where multiple agent controllers are encoded in a single genome (Bongard, 2000; Waibel et al., 2009). In the simplest approach, the different agent controllers are simply concatenated in a single genome (Luke et al., 1998; Suzuki and Arita, 2006). In this case, specific crossover operators can facilitate the exchange of genetic material among different agents (Haynes and Sen, 1997; Lichocki et al., 2013).

Other team learning approaches allow for a dynamic team composition, meaning there is no fixed mapping between an agent and a specific part of the genome. Bongard, (2000), for instance, proposed the *Legion System*, a genetic programming approach where the genome encodes the composition of the team and one program sub-tree for each behaviour class. A different neuroevolution approach was proposed by D'Ambrosio and Stanley, (2008) and D'Ambrosio et al., (2010): all the agent controllers are indirectly encoded in a single genome using compositional pattern producing networks (CPPNs), which are evolved by the *HyperNEAT* algorithm (Stanley et al., 2009). HyperNEAT can exploit similarities in agents' policies, while at the same time allowing for variations.

Concurrent learning

In *concurrent learning* (or *individual encoding*), multiple learning processes for different parts of the team run in parallel. Typically each agent has its own learning process, that modifies its behaviour towards the improvement of the performance of the team as a whole. Each genome thus encodes a single agent controller, which is evaluated together with genomes from the other learning processes. One of the most popular *concurrent learning* algorithms is cooperative coevolution (Popovici et al., 2012; Potter and De Jong, 2000), where the controllers for the different agents are coevolved in separate populations that interact with one another during evaluation.

Another approach that falls under the category of concurrent learning is *online evolution* (Watson et al., 1999), introduced in Section 2.2. Online evolution has specific goals, namely to enable online adaptation and avoid the reality gap, and therefore it is not comparable to static optimization techniques such as cooperative coevolutionary algorithms.

Concurrent learning and team learning each have their advantages and disadvantages. It has been shown that in some conditions concurrent learning is more favourable (Iba, 1996), while on others team learning might be preferable (Miconi, 2003). Concurrent learning is better suited for domains in which some decomposition is possible and helpful, and when it is useful to focus on each sub-problem to some degree independently of the others (Jansen and Wiegand, 2003; Potter and De Jong, 2000). The reason is that concurrent learning projects the larger joint team search space onto separate, smaller individual search spaces. If the problem can be decomposed such that individual agent behaviours are relatively disjoint, then this can result in a dramatic reduction in search space and in computational complexity. Another advantage of concurrent learning techniques is that they facilitate the emergence of agent roles and specialisations (Potter et al., 2001; Yong and Miikkulainen, 2009).

2.4 Cooperative Coevolutionary Algorithms

A coevolutionary algorithm is an evolutionary algorithm (or a collection of evolutionary algorithms) in which the fitness of an individual depends on the relationship between that individual and other individuals (Wiegand, 2003). Such definition naturally implies profound differences with respect to traditional evolutionary algorithms. It can be argued that in coevolutionary algorithms, the individuals are not actually evaluated, but in fact their interactions are evaluated (Wiegand, 2003). The individual fitness evaluations are contextually dependent on the state of other individuals, as an individual represents just a part of the solution, and cannot be evaluated in isolation. Since all individuals are under evolution, however, this means that individual evaluations are subject to a dynamic fitness landscape — the exact same individual can receive a high or low fitness scores depending on the other individuals with which it is evaluated. These algorithms starkly contrast with traditional non-coevolutionary algorithms where the fitness landscape is static — the same individual always receives the same fitness score during the entire evolutionary process.

> **Definition 1. Coevolutionary Algorithm:** An evolutionary algorithm in which the fitness of an individual depends on the relationship between that individual and other individuals (Wiegand, 2003).

The Definition 1, due to its broadness, can arguably encompass single-population evolutionary algorithms where the individuals interact with other individuals of the same population (Popovici et al., 2012). In this thesis, however, we use coevolutionary algorithms to refer exclusively to the more common approach, where there are two or more coevolving populations, and in which individuals are evaluated based on their interactions with individuals from the other population(s).

Among coevolutionary algorithms, the most fundamental distinction is between *cooperative* and *competitive* coevolutionary algorithms (Popovici et al., 2012; Wiegand, 2003). In the case of cooperative algorithms, individuals are rewarded for performing well with the other individuals, and penalised when they perform poorly together. This type of relation is known in biology as mutualism. In the case of competitive algorithms, however, individuals are rewarded at the expense of those with which they interact. That is, individuals are rewarded for outperforming the individuals with which they are competing. In biology, this is known as a predator-prey relationship. The implementation of the two types of algorithms is typically very similar, with the difference residing mainly in the fitness function. The two approaches, however, have very different purposes: while competitive coevolutionary algorithms are used to evolve high-performing *individuals* when there is no absolute notion of fitness (e.g. evolving a chess player from scratch), cooperative coevolutionary algorithms are used to evolve successful *teams* where individuals cooperate to achieve the team's objective (e.g. evolving controllers for a multirobot team that has to complete a given task). In this thesis, we will therefore focus only on cooperative coevolutionary algorithms.

> **Definition 2. Cooperative Coevolutionary Algorithm:** A coevolutionary algorithm in which individuals are rewarded for successful collaborations with individuals from the coevolving populations.

> **Definition 3. Competitive Coevolutionary Algorithm:** A coevolutionary algorithm in which individuals are rewarded for outperforming the individuals in the coevolving populations.

While traditional (single-population) evolution may be applicable to static optimisation problems of arbitrary complexity, the decompositional nature of cooperative coevolutionary algorithms may afford them some advantages for dealing with problems that are complex, but highly structured. Assuming that a suitable problem decomposition can be found or is provided, it is natural that a CCEA could coevolve the various components independently more efficiently than could a traditional EA evolve the entire structure. Since the advent of cooperative coevolutionary approaches, this has been the primary motivating factor for their use (Potter and De Jong, 1994; Potter and De Jong, 2000). One of the common application of CCEAs is the evolution of multiagent behaviours (Popovici et al., 2012; Potter et al., 2001). The natural decomposition of the problem into sub-components makes multiagent systems a good fit for cooperative coevolution: each agent can be represented as a component of the solution, and the coevolutionary algorithm evolves a set of agent behaviours that solve the given task. In this way, CCEAs allow for the synthesis of heterogeneous multiagent systems, where each individual agent can evolve a specialised behaviour.

2.4.1 General Architecture

The classic cooperative coevolution architecture (Popovici et al., 2012; Potter and De Jong, 2000) operates with a system comprising two or more separate populations, meaning that individuals only compete and reproduce with other individuals from their own population. Due to complete separation of populations, it is possible to have different evolutionary algorithms and different individual representations for each population. At every generation of the evolutionary algorithm, each population is evaluated in turn. To evaluate an individual from one population, teams are formed with representative individuals from the other populations. The resulting teams are then evaluated by a fitness function in the problem domain, and the individual being evaluated receives the fitness score obtained by the team as a whole. The fitness differential is thus a function of the individual's contribution to the problem-solving effort within the context of agents from the other populations. Figure 2.4 depicts the basic coevolutionary architecture.

There are a number of choices that need to be made when implementing a cooperative coevolutionary algorithms, concerning the evaluation of the individuals (Wiegand et al., 2001):

Selection of representatives It is typically not computationally feasible to evaluate an individual with all the other individuals in the other populations, as the number of collaborations (teams) that would need to be evaluated would be too high. A subset of individuals for each population therefore has to be chosen, commonly called the *representatives* (see Figure 2.4). Different approaches for choosing the representatives have been studied (Wiegand et al., 2001), including choosing random individuals, choosing the best individuals, as determined by the fitness obtained in the last evaluation, or a combination of the two. Previous works have shown that using the best individuals, or the best plus some random individuals yields the best results (Wiegand et al., 2001).

Number of representatives An ensuing implementation choice is how many representatives should be chosen for each population. Previous works have shown that

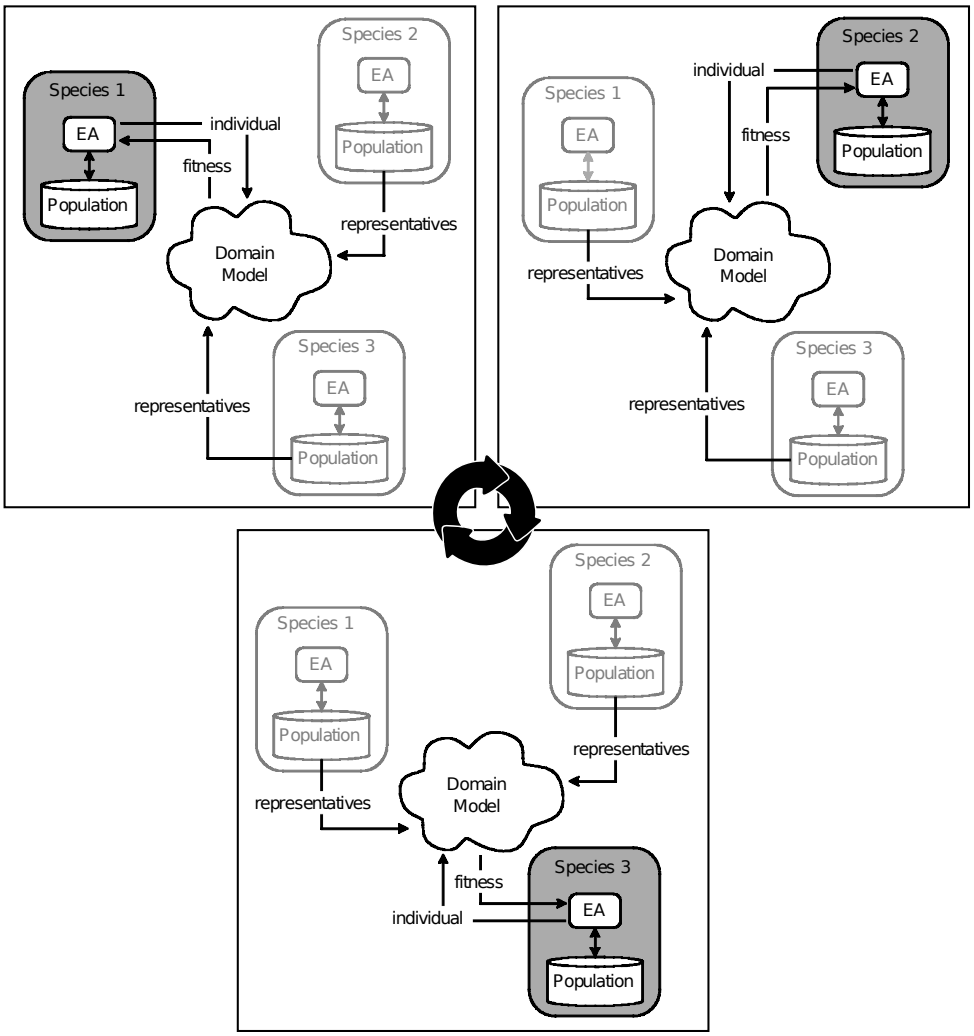


FIGURE 2.4: Cooperative coevolution with three species (populations). In each generation, the individuals of each population are evaluated in turn. The population currently under evaluation is highlighted in grey. Image adapted from (Potter and De Jong, 2000).

increasing the number of representatives can improve convergence to the global optimum (Panait, 2010), but this might not be computationally feasible in many domains, and does not scale with the number of populations. Using only one representative (the previous best) is thus common practice (Popovici et al., 2012).

Combination of evaluations If multiple collaborations are used to evaluate each individual, an ensuing question arises: how to combine these multiple fitness scores to obtain a single score that can be used in the selection and breeding process. Possible approaches include taking the average of fitness scores (hedge), the worst (pessimistic), or the best (optimistic) (Wiegand et al., 2001). Previous works have found that the optimistic reward scheme is generally the best approach (Panait et al., 2003, 2006a; Wiegand et al., 2001), as it is a better estimator of the potential of the individual.

Given the results described above, and the common practice when applying CCEAs to multiagent problems (Popovici et al., 2012), in this thesis we mainly rely on the approach where a single collaboration is used to evaluate each individual,

where the representatives are the individuals that achieved the highest fitness scores in the previous generation. This approach is also known as *single-best collaboration method* (Bucci and Pollack, 2002). The corresponding basic algorithm, which is used throughout this thesis unless stated otherwise, is defined in Algorithm 1.

Algorithm 1 Basic CCEA algorithm used in this thesis.

Let $\mathcal{P}$ be the set of all populations in the coevolutionary system.
for each population $p \in \mathcal{P}$ **do**
    $r_p \leftarrow$ randomly pick one individual $i \in p$
for each generation **do**
    for each population $p \in \mathcal{P}$ **do**
        for each individual $i \in p$ **do**
            $t_i \leftarrow \{i\} \cup \{r_q : q \in \mathcal{P} \wedge q \neq p\}$ ▷ Form one team with the representatives
            $f_i \leftarrow \texttt{Evaluate}(t_i)$ ▷ Obtain the team's fitness
    for each population $p \in \mathcal{P}$ **do**
        $r_p \leftarrow \arg\max_{i \in p} f_i$ ▷ individual $i \in p$ with maximum fitness
        $p \leftarrow \texttt{Breed}(p)$ based on the fitness scores $f_{i \in p}$

2.4.2 Known Limitations and Pathologies

Despite the theoretical potential of cooperative coevolutionary algorithms, applications of CCEAs often fail to achieve the desired results. The reasons for this are mainly associated with the dynamic fitness landscape of cooperative coevolutionary algorithms, which can lead to complex evolutionary dynamics. These limitations are well studied in previous works (Popovici et al., 2012; Wiegand, 2003), and they can compromise the effectiveness of CCEAs. These limitations are the main topic addressed in this thesis, as they represent a serious obstacle to the effective application of CCEAs to complex heterogeneous multiagent tasks. Below, we discuss these limitations in detail.

Premature Convergence to Stable States

In a cooperative coevolutionary algorithm, the fitness landscape of each population is defined (and constrained) by the behaviour of the team members. The fitness landscape is thus constantly changing, as the individuals from the other populations evolve. The fitness of an individual can vary significantly depending on with which collaborators it is evaluated (Wiegand et al., 2001). It is therefore easy for a population to be misled by a particular selection of collaborators from the other populations (Panait et al., 2006b), as the individuals are only rewarded according to their performance with the given collaborators. CCEAs are naturally attracted to Nash equilibria (Wiegand, 2003; Wiegand et al., 2002) where each population is perfectly adapted to one other, such that changing one of the team members would result in a lower team performance.

> **Definition 4. Nash equilibrium:** A joint strategy (one strategy for each agent) such that no single agent has any rational incentive (in terms of better reward) to change its strategy away from the equilibrium.

There is, however, no guarantee that such equilibrium states correspond to globally optimal solutions (Panait, 2010; Wiegand et al., 2002), which is typically the goal

when using evolutionary algorithms. Another consequent issue is *relative overgeneralisation* (Panait et al., 2004; Wiegand and Potter, 2006), which is seen as one of the most prominent challenges in cooperative coevolution (Panait and Luke, 2005a):

> **Definition 5. Relative over-generalisation:** The coevolutionary dynamic that occurs when populations in the system are attracted to regions of the search space in which there are many strategies that perform well with the individuals from the others populations.

Due to these pathological dynamics, it has been shown that, in many cases, CCEAs are actually attracted to suboptimal regions of the search space (Jansen and Wiegand, 2004; Panait, 2010). Premature convergence to equilibrium states should be distinguished from the typical local convergence problems that plague non-coevolutionary algorithms (Panait et al., 2006b). While under ideal conditions a genetic algorithm is theoretically attracted to the global optimum (Rudolph, 1994), the same does not hold for cooperative coevolutionary algorithms. Panait, (2010) has shown that even under ideal conditions of infinite populations, the CCEA might still not be attracted to the optimum. The lack of bias towards optimal solutions can compromise the effectiveness of cooperative coevolutionary algorithms (Panait and Luke, 2005a).

Loss of Fitness Gradients

Loss of fitness gradient occurs when the search gradient for one population suddenly becomes too steep, meaning that it is very unlikely to obtain different fitness scores through random mutations of the current individuals. Although loss of fitness gradients is far more common in competitive coevolution, it can also occur in cooperative coevolution. A classic illustration of loss of gradient in *competitive* coevolution is the situation were a chess Grand Master plays against a child. If the child receives no information other than the outcome of the game, the child has almost no means of learning how to improve her game. A loss of gradient can occur in this competitive setting when one population suddenly achieves a level so superior to the other, that nothing can be learned by either population by competing.

> **Definition 6. Loss of fitness gradient:** The coevolutionary behavior that occurs when one population or group reaches a state such that other groups and populations lose necessary relative fitness diversity from which to continue meaningful progress (Wiegand, 2003).

An example of loss of fitness gradient in *cooperative* coevolution might be, for instance, a team of two players learning how to play Pictionary.[1] If the player that is guessing has no ability at all, and can only do random guesses, the player that is drawing does not have enough feedback to improve his drawing skills. The skills of the drawing player thus become irrelevant (the gradient is lost), as they alone are not enough to make any impact in the team's performance. Synchronised learning and mutual development of skills are therefore essential in a cooperative coevolutionary algorithm (Uchibe et al., 1998).

[1]The Pictionary game is played with teams of two, with players trying to identify specific words from their teammates' drawings. Both good drawing skills and as good guessing skills are required to be a successful team.

Scalability

The classic CCEA architecture is associated with inherent scalability issues with respect to the number of agents in the team. When each agent is evolved in a separate population, increasing the number of agents can lead to three issues: increase of computational complexity (Potter and De Jong, 2000), redundant learning (D'Ambrosio et al., 2010), and the credit assignment problems (Agogino and Tumer, 2008; Colby and Tumer, 2015b).

Computational complexity In heterogeneous multiagent systems, each agent of the system is typically co-evolved in a separate sub-population. As such, the number of sub-populations increases linearly with the number of agents. The computational complexity thus grows with the number of populations, both in space and time, due to the increased number of individuals to evaluate each generation (Potter and De Jong, 2000).

Nitschke et al., (2012a), for instance, showed that the CONE method can be effectively used to evolve teams of up to 100 agents, each with its own unique controller, i.e., 100 populations. The computational complexity was, however, overwhelming: each population was initialized with 400 genotypes, which resulted in a total of 40.000 genotypes and 400.000 evaluations per generation (each genotype was evaluated 10 times). Considering that in multirobot domains the evaluations typically require the simulation of the whole system performing the task, which can be computationally expensive in itself, this results in a computational cost that is infeasible for most applications.

Redundant learning Since agents are evolved in isolated populations, they must separately discover all aspects of the solution, even though there may be a high degree of overlapping in the resulting policies of each agent. This problem is often known as *redundant learning*, or the *problem of reinvention* (D'Ambrosio et al., 2010; Panait and Luke, 2005a). In a typical collective robotics tasks, robots tend to share a basic skillset, such as basic navigation or obstacle avoidance. In large multirobot teams this becomes even more evident, as many robots in the team can share entire behaviour policies (Nitschke, 2012). Most CCEAs, however, do not cope with this peculiarity, as there is a strict division between the coevolving populations. The evolutionary process can thus potentially waste many resources learning the same behaviour in different sub-populations. Potter et al., (2001) address this problem by pre-programming the shared skillset in the robots, and the evolved robot controllers operate only with high-level actions. A distinct approach to avoid this problem is the implementation of a *shaping phase* before the coevolutionary algorithm. The shaping phase (Nitschke et al., 2012a) evolves a single agent controller that possesses the basic skillset needed for the task. The initial populations of the coevolutionary algorithm are then formed by replicating and mutating the pre-evolved agent controller.

Credit assignment Credit assignment issues (Agogino and Tumer, 2008; Colby and Tumer, 2015b; Rahmattalabi et al., 2016) can occur in large teams, when the impact of a single agent on the performance of the whole team becomes almost inconsequential, thus causing the fitness gradients to vanish (Agogino and Tumer, 2008). This is essentially a signal-to-noise problem, where the selective signal for an agent's behaviour is drowned by the noise of all the other agents' impact on the evaluation function.

2.4.3 Extensions of the Basic Architecture

The best-known extensions of the cooperative coevolution architecture are described next. These extensions aim at better suiting the CCEA algorithm to certain classes of problems, overcoming some of the limitations discussed above.

Multiagent Enforced SubPopulations (MESP)

Enforced SubPopulations (ESP) (Gomez and Miikkulainen, 1997) is a neuroevolution method that allocates a separate population to each hidden unit in the network, where the individuals of the subpopulations encode the incoming and outgoing connection weights of the respective neuron. To assess the quality of each individual, collaborations are established with individuals (neurons) randomly chosen from the other sub-populations. The full network is then evaluated in the domain, and the fitness is shared among the participating neurons.

Yong and Miikkulainen, (2009) extended ESP to the evolution of multiagent systems, proposing Multiagent ESP (MESP). MESP is based on the cooperative coevolution architecture of Potter and De Jong, (2000). Each coevolving population evolves the neural network of one agent, using the Enforced SubPopulations (ESP) neuroevolution algorithm. The networks are then evaluated together in the task as a team, and the resulting fitness is distributed among those networks. In this way, MESP enables two levels of cooperation: the neurons are required to cooperate to form neural networks, and these networks must cooperate in a team in order to receive a high fitness score.

COllective NeuroEvolution (CONE)

COllective NeuroEvolution (CONE) (Nitschke, 2008; Nitschke et al., 2010) is an extension of Multiagent ESP (Yong and Miikkulainen, 2009). As in MESP, each population is composed by several sub-populations, each representing one hidden neuron of a neural network. The distinctive characteristic of CONE, when compared to other coevolutionary methods, is that CONE implements mechanisms for regulated combination between and within populations. CONE includes the following unique features:

Genotype difference metric (GDM): A heuristic that regulates recombination of similar genotypes in different populations. Two genotypes can only be recombined if they are considered similar, i.e, if the average weight difference between them is below a dynamic threshold.

Specialisation difference metric (SDM): A heuristic that regulates genotype recombination based on behavioural similarities exhibited by full controllers. If the fittest controllers of two populations have similar behaviour specialisations, then these two populations go through a recombination process. The GDM is applied to regulate the recombination of similar genotypes within the two populations.

Controller size adaptation: A heuristic that adapts the number of hidden layer neurons in each controller over the course of cooperative co-evolution. If fitness stagnation is detected in one population, a new hidden neuron is added.

The degree of specialisation of each controller is given by the number of times the robot switches between different actions, relative to the maximum number of

possible switches. If the degree of specialisation is above a certain threshold, the controller is considered specialised in action x, where x is the action in which the controller spent most time. Otherwise, the controller is considered unspecialised. A natural limitation of CONE is that for this mechanism to work, the actions of the agents need to be clearly distinguishable from one another. The possible specialisations of each controller are specified manually by the experimenter. CONE has been shown to effectively encourage the specialisation of the agents in the team and reduce redundant learning. A significant amount of task-specific knowledge is, however, required in order for CONE to work.

Evaluation with Informative Collaborators

With the objective of improving convergence to the global optimum, a number of strategies have been proposed in which multiple collaborations are used to evaluate each individual. The rationale is to reduce the variability of individual evaluations, so that there is a more accurate estimate of the individual's absolute worth, with less sensitivity to the evaluation context. This type of strategies naturally come at a cost of increased computational complexity, which might help explain why they have only been studied in function optimisation benchmarks (in which the computational cost of an evaluation is minimal), typically with only two populations. Some of these strategies are detailed below.

Optimistic reward scheme: Panait et al., (2004), Popovici and De Jong, (2005), and Wiegand et al., (2001) demonstrated that an optimistic reward scheme can be used to bias coevolution towards globally optimal solutions. The optimistic scheme evaluates an individual, not with only one collaboration, but in N trials, each with a randomly formed collaboration, and only the maximum reward obtained is considered. Panait, (2010) showed that this optimistic scheme guarantees convergence to a global optimum if given enough resources, i.e., sufficiently large populations, and a sufficiently high number of collaborations N. To reduce the number of necessary evaluations, Panait and Luke, (2005b) proposed a variation of the optimistic reward scheme, wherein the number of collaborations N decreases throughout the evolutionary process.

iCCEA: Panait et al., (2006a) presented an archive-based algorithm called *iCCEA*, in which the number of evaluations is reduced by maintaining an archive of *informative collaborations* for each population. *iCCEA* builds an archive of collaborators which produce the same ranking of individuals in the other population as they would receive if they were tested against the full population of collaborators. The authors, however, acknowledge that the archive can become large, which makes evaluation computationally expensive, and that it is unclear if/how the algorithm would scale to complex domains.

Biased CCEA: The maximum of N collaborations scheme was extended by Panait et al., (2006b): the fitness is based partly on the maximum score obtained in N collaborations with randomly chosen partners, and partly on the reward obtained when partnering with the *optimal* collaborator, i.e., the collaborator with which the individual under evaluation would receive the highest possible fitness score. The results showed that computing the fitness of an individual based on its performance with the optimal collaborator can significantly increase the performance of the algorithm. The assumption that the optimal collaborator is known is, however, largely unrealistic for most domains, and

as such heuristic methods would be necessary for estimating the optimal collaborator, such as relying on the history of highest-scoring individuals.

mCCEA: A new approach for increasing convergence to global optima was proposed by Peng et al., (2016), named *Multi-population Mechanism Based CCEA* (mCCEA). In mCCEA, each population is composed of a base and several child populations. These child populations conduct dynamic multimodal optimization so as to obtain local and global optima as the representative collaborators with high fitness and diversity. The child populations evolve with a local search algorithm, focusing on disjoint parts of the search space, while the base population evolves with the typical genetic algorithm. The representatives individuals of each population are the highest-fitness individuals of each of its child populations. mCCEA was evaluated in several benchmark problems of function optimisation, with two populations. It was able to significantly outperform the *Biased CCEA* and *Optimistic reward scheme* presented above.

Difference Evaluation Functions

Agogino and Tumer, (2008) proposed an alternative approach to facilitate convergence to (near-)optimal solutions, relying on the use of difference evaluation functions. The difference evaluation function is a shaped reward signal that provides agent-specific evaluation by removing a large amount of the noise created by the actions of other agents in the system. When a team of agents is evaluated, the fitness of each agent is calculated with a difference evaluation (the agent's contribution to the team), rather than the global system evaluation (the team's performance). The difference evaluation of a given agent is defined as the difference between the team's performance with the agent and the team's performance without the effects of that agent. It has been shown that difference evaluation can significantly outperform the standard global evaluation, in both evolutionary game-theory (Colby and Tumer, 2015a) as well as multirobot tasks (Colby and Tumer, 2015b). The limitation of this approach is that it is not always clear how to effectively compute the difference, i.e., how to evaluate the performance of the team without the effects of a given agent. In tasks where simply excluding the agent from the team is not feasible, it might be necessary to manually provide a *default* behaviour for each of the agents (Colby and Tumer, 2015b; Yliniemi and Tumer, 2016).

2.4.4 Domains of Application

In this section, we review the multiagent problems that have been addressed with cooperative coevolutionary algorithms.

Predator-prey Pursuit

This is one of the most common tasks in multiagent coevolution research (both cooperative and competitive). Cooperative pursuit games consist of a number of agents (predators) cooperatively chasing a prey. Individual predator agents are usually not faster than the prey, and often agents can sense the prey only if it is close by. Therefore, the agents need to actively cooperate in order to successfully capture the prey. In cooperative coevolution studies, typically only the team of predators is evolved, while the prey is given a fixed pre-programmed behaviour. The task is interesting because heterogeneity in the predator team is required to effectively catch the prey (Yong and Miikkulainen, 2009).

- Nitschke et al., (2012b) used CONE to evolve a team of 2–6 predators to capture 1–2 preys in a bounded environment. In this variant of the task, the predators could sense obstacles, other predators, and preys. Each predator was controlled by a low-level neural network that received the sensor values and output the wheels speed. CONE was compared to MESP and CCGA (Cooperative Coevolution Genetic Algorithm).

- Yong and Miikkulainen, (2009) used MESP with incremental evolution to evolve a predator team of three robots. The neural network of each predator received the position offset of the prey and outputs the direction in which the agent should move. The authors studied the impact of communication between predators, the heterogeneity of the team, and the necessity of coevolution.

- Blumenthal and Parker, (2004) evolved a team of four predators (vs one prey) using Punctuated Anytime Learning (Parker and Blumenthal, 2002). The environment was open, and the predators had to catch the prey before it escaped the arena. Each predator had different movement capabilities, differing in the turning rate and maximum speed.

- Rawal et al., (2010) used the same experimental setup as (Yong and Miikkulainen, 2009). However, in this study the prey was also coevolved with the predators. While the predator learn to cooperate to catch the prey, the prey learns to evade them. The experiments showed that it is possible to sustain coevolution of teams of competing and cooperating agents.

- In the work reported in the Chapter 3 of this thesis and in (Gomes et al., 2014a, 2017b), we present experiments where we evolve control for heterogeneous teams of two to seven predator robots. Only one pre-programmed prey was present, similar to the setup in (Yong and Miikkulainen, 2009).

- The predator-prey task is also used in the real-robot experiments described in Chapter 4 and (Gomes et al., 2016b).

Herding

Potter et al., (2001) proposed a herding task in which a group of robots (the shepherds) has to force another robot (the sheep) into the corral. The herding environment consists of a pasture that is fenced on three sides, with a smaller enclosed corral on one of the fenced sides. The sheep tries to avoid the corral, trying to escape through the unfenced side of the pasture. To make the task more challenging, there can also be a predator robot (the fox) that attempts to catch the sheep. In (Potter et al., 2001) the controllers of the shepherds, under evolution, were high-level neural networks. The study focused on the topic of heterogeneity vs homogeneity in teams of cooperating robots.

The herding task was also used in some of the preliminary work developed for this thesis. In these experiments (Gomes et al., 2015b), which are not reported in this thesis, we used different versions of the herding task, varying the number of shepherds, foxes, and sheep.

Collective Construction

The *gathering and collective construction* task (Nitschke et al., 2012a) requires that robots search for building blocks in the environment, transport them to a construction zone, and place them in a specific sequence of block types required for structure assembly. There are two different types of blocks, and the robots must choose between different sensor settings in order to detect the blocks of a given type. Specialisation is thus beneficial for the efficient solution of the task. Large teams were evolved (50 and 100 robots) using CONE, MESP, and the traditional CCEA. Incremental evolution was used to bootstrap the coevolutionary algorithm.

Rover Problem

In the *rover problem* (Agogino and Tumer, 2004, 2008; Colby and Tumer, 2015b), a collective of rovers on a two dimensional plane aims to observe points of interest (POIs) scattered throughout the environment. Each POI has an associated value, and each observation of a POI made by a rover yields an observation value that is proportional to the proximity of the rover to the POI. The POI locations are static throughout the experiment. The objective of the rovers is to maximise the observation values of the POIs over the course of an episode. An increasing system evaluation corresponds to better observation coverage of the POIs. In (Colby and Tumer, 2015b), the multi-rover problem was solved using a standard CCEA and other algorithms based on difference evaluation functions.

Collective Foraging

Nitschke et al., (2010) proposed an *extended multi-rover* task, which is an instance of collective foraging tasks commonly found in swarm robotics studies (Bayındır, 2016). This task requires a team of simulated autonomous vehicles (named *rovers*), to detect and collect features of interest (named *red rocks*) with a maximal total value over the course of the team's lifetime. A red rock is collected when it is within range of more than one rover's red rock detection sensors. Furthermore, there are five types of red rocks, and three different resolution settings for red rock detection sensors. For each type of red rock, specific combination of sensor settings are required to detect it. The environment is rich in walls and obstacles. In (Nitschke et al., 2010), a team of 20 robots was evolved, using CONE, MESP, and a standard CCEA. Incremental evolution was used to bootstrap the coevolutionary algorithm.

In this thesis, we use similar collective foraging tasks for a number of different experiments. In Chapter 3, we use a version with two rovers, where both of them are simultaneously needed to collect each item (see also Gomes et al., 2017b); in Chapter 5, we use a task where an aerial and ground robot have to cooperate to find and collect items (see also Gomes et al., 2016a); and in Chapter 6, we use a version similar to the *extended multi-rover* task, where there are different item types and robots can have different sensor resolutions.

Soccer Games

Simulated soccer games are widely popular in multiagent learning, and remain a considerable challenge regarding the control of the teams (Barrett and Stone, 2015). In preliminary work (Gomes et al., 2014a), not reported in this thesis, we solved

a keepaway soccer task (Stone et al., 2005) using cooperative coevolutionary algorithms to evolve the controllers of the keepers. Keepaway soccer is a simplified version of robot soccer in which there are usually three keepers and one or two takers. The keepers must learn to keep possession of the ball against a taker that actively tries to snatch it from the keepers.

In Chapter 6, we present experiments with a full simulated soccer task, where teams of five agents play against each other, trying to score on the opponent's goal.

Non-embodied Multiagent Problems

A number of studies focus instead on non-embodied agents – software agents or pseudo-agents that simply encode an element of the solution. Although our thesis focus on embodied and autonomous agents, such as robots, looking at these applications is valuable to understand the potential and broad applicability of coevolutionary algorithms.

- Several studies have applied CCEAs to static function optimisation, where each agent represents one parameter of the function (Panait, 2010; Panait et al., 2006b; Potter and De Jong, 1994; Wiegand et al., 2001). These problems have mostly been used for benchmark and demonstration purposes, and do not have any direct practical application.
- Soria et al., (2016) demonstrates an application of CCEAs to the industrial design of a racing car, where each agent encodes a set of parameters that define a given body part (e.g., an agent encodes the parameters of the engine, other agent defines the rear tires, etc.).
- Agogino and Tumer, (2007) employ CCEAs to solve a problem of air traffic flow management. Each agent, under evolution, represents a ground location throughout the airspace, and it is responsible for any aircraft going through it.

2.5 Evolution Driven by Behavioural Diversity

Traditionally, evolutionary robotics revolved around applying standard evolutionary algorithms to robotics problems (see Section 2.2): the experimenter defines the fitness function, which establishes the task's goals, and the evolutionary process runs until it converges. Unfortunately, due to the high complexity and rugged fitness landscapes of robotics tasks, evolutionary algorithms often fail when faced with more challenging problems (Silva et al., 2016b). These issues have traditionally been mitigated by introducing additional task-specific biases in the fitness function (Doncieux and Mouret, 2014; Nelson et al., 2009), avoiding local optima but at the same time introducing a great deal of experimenter-induced bias in the evolutionary process, which is contrary to the ultimate purpose of evolutionary robotics (Eiben, 2014).

The field of evolutionary robotics has recently started to shift to alternative techniques that go the opposite way: instead of focusing and driving the evolutionary search towards a certain solution, they instead promote behavioural exploration and novelty (Doncieux and Mouret, 2014; Doncieux et al., 2015; Silva et al., 2016b). As Doncieux and Mouret, (2014) argue in a recent survey: *interesting results can be generated when evolutionary robotics is not considered purely as black-box optimization*. In this thesis, we employ novelty-driven approaches and take into account the exploration of the behaviour space to mitigate fundamental issues in the cooperative coevolution of heterogeneous multiagent systems.

2.5.1 Premature Convergence and Deception

Evolutionary algorithms are prone to suffer from deception (Jones and Forrest, 1995; Whitley, 1991), a challenging issue in evolutionary computation that causes the evolutionary process to converge prematurely to local optima. Deception occurs when the fitness function creates a *deceiving* fitness gradient. This typically happens when the fitness function fails to adequately reward the intermediated steps that are needed to achieve the global optimum. In the case of coevolutionary algorithms, deception is also present, but can have different causes. As discussed in Section 2.4.2, in CCEAs, the fitness of an individual is dependent on other individuals – the team members. Therefore, individuals can be deceived by the fitness gradient induced by the collaborators with which they are evaluated (Panait et al., 2004).

In non-coevolutionary algorithms, deception can be mitigated through the use of techniques that maintain genotypic diversity in the population. Such techniques include fitness sharing (Goldberg and Richardson, 1987), promotion of diversity based on the fitness score of the solutions (Hu et al., 2005; Hutter and Legg, 2006), intermingling individuals of different genetic ages (Castelli et al., 2011; Hornby, 2006), and minimisation of the age of the genotypes (Schmidt and Lipson, 2011). Other techniques to overcome deception rely on the decomposition of the objective into multiple sub-goals that are easier to attain. These techniques include incremental evolution (Gomez and Miikkulainen, 1997; Mouret and Doncieux, 2008), fitness shaping (Uchibe et al., 2002), and multi-objectivisation of sub-goals (Knowles et al., 2001; Mouret and Doncieux, 2008; Trianni and López-Ibáñez, 2015).

2.5.2 Novelty Search

While the methods discussed above for mitigating deception might help the evolutionary process to avoid getting stuck in local optima, they leave the underlying problem untreated: the fitness gradient itself might be misdirecting the search. With this issue in mind, a new evolutionary approach was proposed — novelty search (NS) (Lehman and Stanley, 2011a). Novelty search drives evolution towards behavioural novelty instead of a pre-defined goal. The distinctive aspect of novelty search is how the individuals of the population are scored. Instead of being scored according to how well they perform a given task, which is typically measured by a static fitness function, the individuals are scored based on their behavioural novelty according to a dynamic *novelty metric*, which quantifies how different an individual is from other, previously evaluated individuals. This reward scheme therefore creates a constant evolutionary pressure towards behavioural innovation, and actively tries to avoid convergence to a single region in the solution space.

Implementing novelty search requires little change to any evolutionary algorithm aside from replacing the fitness function with a domain-dependent novelty metric (Gomes et al., 2015c; Lehman and Stanley, 2011a). To measure how far an individual is from other individuals in behaviour space, the novelty metric relies on the average behaviour distance of that individual to the k-nearest neighbours:

$$\rho(x) = \frac{1}{k}\sum_{i=1}^{k} \mathrm{dist}(x, \mu_i) \ , \tag{2.1}$$

where μ_i is the ith-nearest neighbour of x with respect to the distance metric *dist*. Potential neighbours include the other individuals of the current population and a sample of individuals from previous generations, stored in an archive. The purpose

of the archive in novelty search is to encourage exploration of new behaviour regions, besides maintaining diversity in the population. Without the memory effect provided by the archive, evolution may cycle between behaviour regions, due to the lack of evolutionary pressure towards novel regions of the search space (Gomes et al., 2015c; Lehman and Stanley, 2011a). The function *dist* is a measure of *behavioural* difference between two individuals, which will be further discussed in the next section. It should not be confused with the genotypic distance commonly used for speciation in fitness sharing techniques (Goldberg and Richardson, 1987).

In novelty search, candidate solutions from sparse regions of the behaviour space thus tend to receive higher novelty scores, which results in an evolutionary process that strives to uniformly explore the behavioural space. This dynamic is illustrated in Figure 2.5, where a maze navigation task is solved with both novelty search and fitness-based evolution. As the novelty metric promotes behavioural diversity within the population at all times, it avoids convergence to a single point in the solution space, which is common in fitness-based evolution.

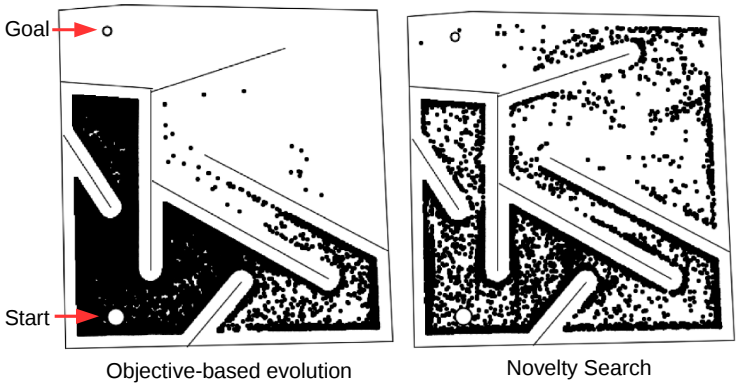


FIGURE 2.5: Behaviour exploration of fitness-based evolution and novelty search in the deceptive maze task. A robot has to navigate from the starting point to the goal point. The dots represent the final position of each evolved individual. Fitness-based evolution often gets trapped in a local optimum, while novelty search explores the search space more uniformly. Image adapted from (Lehman and Stanley, 2011a).

Novelty search has mainly been studied in the evolutionary robotics domain, including the evolution of: (i) single-robot controllers (Mouret and Doncieux, 2012), (ii) robot gait control (Lehman and Stanley, 2011a), (iii) controllers for multirobot and swarm robotic systems (Gomes et al., 2013), (iv) robotic morphologies (Lehman and Stanley, 2011b), and (v) plastic neural networks (Risi et al., 2010). A number applications of novelty search outside the robotics domain can also be found in the literature, for instance in game content generation (Liapis et al., 2015), design of electrical circuits (Naredo et al., 2016), and machine learning, including regression (Martínez et al., 2013), clustering (Naredo and Trujillo, 2013), and classification (Naredo et al., 2013). The previous works have shown that novelty search is able to find good solutions faster and more consistently than fitness-based evolution in many different applications. Novelty search is particularly effective when dealing with deceptive domains, and can be further combined with fitness-oriented evolution to balance exploration with exploitation (Lehman et al., 2013), which will be discussed in Section 2.5.5.

2.5.3 Configuring the Novelty Search Algorithm

Besides the definition of the behaviour distance metric, there are other algorithmic choices that must be taken into consideration when implementing novelty search. In a recent work (Gomes et al., 2015c), which is not fully reported in this thesis, we conducted a comprehensive empirical study on the parameters and configuration choices of novelty search. These experiments were conducted with a standard single-population non-coevolutionary algorithm, since it is the most common usage of novelty search. Our study was based on the simulated maze navigation task, a single-robot task also extensively used in novelty search studies (Lehman et al., 2013). We used a set of different task setups that confronted novelty search with varying levels of deception and difficulty in exploring the behaviour space. We analysed the results in two dimensions: (i) whether an evolutionary run is able to produce an effective solution or not; and (ii) whether novelty search is able to explore the behaviour space thoroughly and uniformly, regardless of the objective. The main findings are summarised below.

Number of nearest neighbours The number k of nearest neighbours used in computation of novelty scores, see Equation 2.1. We experimented with values ranging from 1 (only the nearest neighbour is taken into account) to k = population size (the entire population can be used to compute the novelty score). Our results showed that the parameter k is robust to moderate variation, but the optimal value depends on the type of archive used. Low to medium values of k were generally preferable. A value of k=15, which is used in a large number of novelty search studies (Gomes et al., 2015c), yielded relatively good performance across all the tested archive types.

Archive of past individuals Deciding on which individuals should be added to the archive, how many of them, and even if the archive is needed at all, has been a topic of discussion. We compared three common approaches for composing the archive: (i) no archive is used, only the current population; (ii) every generation the most novel λ individuals are added to the archive; and (iii) every generation, λ randomly chosen individuals are added to the archive. Our results showed that a randomly composed archive is preferable over a novelty-based one, yielding better results across all the considered metrics. Moreover, our results showed that novelty search is robust to moderate variations of the archive growth rate (λ).[2]

Underlying EA mutation rate An open question was how the degree of genetic diversity influences the behavioural diversity and novelty generated in the novelty search process. To study this effect, we focused on the mutation rate of the EA, which is closely associated with the degree of genetic diversity. We studied the mutation rate in the NEAT neuroevolution algorithm (Stanley and Miikkulainen, 2002) and in a direct-encoding genetic algorithm. Our experiments showed that high mutation rates did not cause better exploration of the behaviour space. Novelty search actually benefited from lower mutation rates, when compared to fitness-based evolution.

[2]Note that the computational cost of the nearest-neighbours calculation increases linearly with the size of the population and the size of the archive. However, it is possible to limit the size of the archive and to use data structures such as KD-trees to reduce this cost.

2.5.4 Behavioural Distance Measures

To quantify the behaviour distance between two individuals, the behaviour of each individual is typically characterised by a real-valued vector – the *behaviour characterisation* (BC). The behaviour distance *dist* is then the distance between the corresponding characterisation vectors. The design of a behaviour characterisation has direct implications on the effectiveness of novelty search. An excessively detailed characterisation can open the search space too much, and might cause evolution to focus on regions of the behaviour space that are irrelevant for solving the task (Cuccu and Gomez, 2011). On the other hand, an incomplete or inadequate characterisation can cause counterproductive conflation of different behaviours (Kistemaker and Whiteson, 2011). Conflation occurs because the mapping between observable behaviours and behaviour characterisations is typically not injective. As such, notably different behaviours can have similar behaviour characterisations, which can potentially hinder the evolution of novel solutions (Kistemaker and Whiteson, 2011).

Task-Specific Behaviour Characterisations

Most previous works on behavioural diversity rely on behaviour characterisations designed specifically for the given task. These characterisations are composed of behavioural traits that the experimenter considers relevant for describing agent behaviour in the context of the given task. For instance, in a maze navigation task, it can be the final position of the robot (Lehman and Stanley, 2011a); in a multirobot aggregation task, it can be the mean distance to the centre of mass (Gomes et al., 2013). Based on the previous works, we identified a number of similarities in task-specific behaviour characterisations (Gomes et al., 2014e):

- The behaviour features are typically related to the fitness function, in the sense that solutions with very different fitness scores typically have distant characterisations. The opposite relation often does not hold – solutions with similar fitness scores can have distant characterisations.
- There is a strong focus on the spatial relationships between entities in the task environment, or the location of the robots in the environment.
- Characterisations typically comprise only a small number of different behavioural traits (up to four).
- Most characterisations focus either on the final state of the environment, values averaged over an entire trial, or a single feature sampled over time.

While designing behaviour characterisations tends to be relatively straightforward and does not require fine-tuning, they do introduce an additional experimenter bias in the evolutionary process, which might not be desirable. To overcome this issue, a number of generic behaviour characterisations have been proposed.

Generic Behaviour Characterisations

Gomez, (2009) was the first to propose the use of task-agnostic generic behaviour characterisations for assessing the behavioural distance. In the proposed approach, action records for the agents are compared, using either the Hamming distance, relative entropy, or normalised compression distance (NCD). The study was conducted with a single-agent discrete task. Doncieux and Mouret, (2010) and Mouret

and Doncieux, (2012) extended generic measures to evolutionary robotics. The proposed measures are applicable to single-robot tasks and are exclusively based on the sensor-effector states of the agent. They rely on comparisons between the sequences of all binary sensor and effector values of the agent through time, or counting how many times the agent was in each possible sensor-effector state. The following measures were proposed:

Hamming distance: Distance between the sequence of all the binary sensor and effector values of the agent sampled through time.

DFT: A discrete Fourier transform is applied to the sensor-effector sequence, and the first coefficients are used to compute the distances between individuals.

State count: Each possible sensor-effector state corresponds to one entry in a vector. Each entry contains the number of times the corresponding state was visited.

These generic characterisations were extended by us in (Gomes and Christensen, 2013), making them applicable to multiagent systems, and to non-binary sensors and effectors. We proposed two generic characterisations applicable to both single and multirobot systems:

Combined state count: Sensor-effector states are defined based on a discretisation of the values from the sensors and effectors recorded at each individual robot. The number of times the robots are in each state is then aggregated, with no discrimination regarding which robot was in a particular state, thus obtaining the behaviour characterisation of the group. We proposed techniques to efficiently represent the sensor-effector states and compute the distance between characterisations, based on filtering mechanisms and the hashing of the states.

Sampled average state: Uses the full history of the sensor-effector states of each robot through time. As such, it contains a temporal component that is not present in the *combined state count* measure. The state of the group at a given instant is the average of the sensor-effector states of all robots, which allows scalability in respect to the size of the group. Additionally, the state of the group is averaged over time windows of equal length (for example, beginning of the simulation, middle, and end), which reduces the sensitivity to the initial conditions and to the stochastic nature of the individual robots' behaviour. The behaviour characterisation of the group is the average sensor-effector state of the robots in each of the time windows.

While generic characterisations are widely applicable, they can result in a very large behaviour space (Cuccu and Gomez, 2011; Mouret, 2011). To address this concern, we proposed and studied a middle ground between generic and task-specific characterisations: *systematically derived behaviour characterisations* (SDBCs) (Gomes et al., 2014e). The proposed measures are directly derived from a formal description of the task state. This way, we can reduce the dependency on the experimenter's knowledge about the task while, at the same time, obtain characterisations that are directly related to the task. The behaviour features that are automatically extracted from the formal description of the task include average distances between the agents, distances between the agents and the environment objects, and the average internal state of the agents, such as speed, energy levels, and so on. Optionally, a set of feature weights can also be calculated, using the characterisations of the current population individuals. The weights are calculated based on the mutual information between the feature values and fitness scores, thus estimating the relevance of each feature for the solution of the task.

Meyerson et al., (2016) proposed a new approach for learning behaviour characterisations, building on the generic measures based on sensory-effector states, and on the weighting scheme proposed by SDBCs. This new approach aims at learning characterisations that can be used across many tasks within the same domain. Generic characterisations are composed of features that represent the probability of the agent taking a particular action in a particular state. The algorithm then learns a weighting vector for these features, using several different training and test tasks. While the learning process is computationally expensive, this approach can be valuable for domains that have many related tasks that need to be solved.

In a separate effort for minimising the experimenter bias, Doncieux and Mouret, (2013) showed that different similarity measures (generic or task-specific) can be combined, either by switching between them throughout evolution or by calculating the behaviour distance based on all similarity measures. It is shown that randomly switching between multiple similarity measures improved performance over any single measure.

2.5.5 Combining Exploration with Objectives

It has been shown that novelty search can struggle to find good solutions when the behaviour space is vast (Cuccu and Gomez, 2011; Gomes et al., 2015c; Lehman and Stanley, 2010), as a great effort might be spent exploring regions that are irrelevant for the task objective. This problem is typically overcome by combining the exploratory pressure of novelty search with the exploitative character of fitness-based evolution. Such combination can lead to a more effective evolutionary process (Lehman et al., 2013), where solutions can be reached faster and more consistently, with a relatively low impact on the diversity of behaviours explored.

A number of techniques have been proposed to accomplish this combination. The first class of techniques relies on a minimal criterion that the individuals must meet in order to be considered viable for selection. This minimal criterion can either be static and provided by the experimenter (*MCNS* – Minimal Criteria Novelty Search) (Lehman and Stanley, 2010), or dynamic and calculated based on the fitness scores of the current population (*PMCNS* – Progressive MCNS, Gomes et al., 2012, 2014d). Liapis et al., (2015) uses a different approach where two populations are used: one contains feasible individuals, which are scored based on novelty, and the other contains infeasible individuals, which are scored based on their proximity to the feasibility threshold.

The second class of techniques bases its selection process on novelty and fitness scores simultaneously. Mouret, (2011) proposed novelty-based multi-objectivisation, where a novelty objective is added to the task objective (fitness function) in a Pareto-based multi-objective evolutionary algorithm (MOEA). A simpler multi-objectivisation is proposed by Cuccu and Gomez, (2011), where the score of each individual is based on a *linear scalarisation* of its novelty and fitness scores, allowing the experimenter to control the relative weight of the novelty and fitness scores. In (Inden et al., 2013), half of the population is subject to novelty-based selection, while the other half is subject to fitness-based selection.

In the empirical study reported in (Gomes et al., 2015c), we compared different combination techniques, including PMCNS, linear scalarisation with different weights, and multi-objectivisation. The highest performing methods for combining novelty and fitness were the multi-objectivisation of novelty and fitness scores, and the linear scalarisation with an equal weight for novelty and fitness, with no major differences between these two.

An alternative to combine exploration with objectives are *Quality Diversity* (QD) algorithms (Pugh et al., 2016) (also known as *Illumination Algorithms*, Mouret and Clune, 2015). QD algorithms are unique in the sense that they are not intended to overcome premature convergence, but rather to discover a repertoire of high-quality solutions located in different regions of the behaviour space. One of the first QD algorithms was *Novelty Search with Local Competition* (NSLC) (Lehman and Stanley, 2011b), in which a multi-objective algorithm combines the novelty score with a local competition objective, thereby encouraging the evolution of a diverse set of high-quality solutions. The more recent MAP-Elites algorithm (Cully et al., 2015; Mouret and Clune, 2015) divides the behaviour space into discrete bins, and aims at finding high-quality solutions in each bin of the behaviour space.

2.6 Summary

In this section, we began by reviewing the state of the art in heterogeneous multi-robot systems. While this is a field that its still in its infancy, the existing studies confirm the potential of such systems for a variety of real-world problems. Synthesising control for heterogeneous systems is currently a challenge, due to the increased search space that comes with heterogeneity. We presented cooperative coevolutionary algorithms as a promising solution for evolving control for such systems. The review of the state of the art, however, reveals that cooperative coevolutionary algorithms still face serious limitations. We identified and discussed the main issues, namely premature convergence to mediocre stable states and poor scalability with respect to number of agents, and described how these issues have been approached in previous studies. In the rest of this thesis, we propose and study methods that deal with these fundamental issues of CCEAs, bringing to coevolutionary algorithms concepts that have showed considerable successes in non-coevolutionary algorithms, such as evolution driven by behavioural exploration, and dynamic team heterogeneity.

Chapter 3

Overcoming Premature Convergence

In this chapter, we study how novelty search can be used to avoid the counterproductive attraction to stable states in cooperative coevolution (Panait et al., 2004; Wiegand and Potter, 2006). Novelty search (Lehman and Stanley, 2011a) is an evolutionary approach that drives evolution towards behavioural novelty and diversity, rather than exclusively pursuing a static objective. We evaluate three novelty-based approaches that rely on, respectively (i) the novelty of the team as a whole, (ii) the novelty of the agents' individual behaviour, and (iii) a combination of the two. We compare the proposed approaches with traditional fitness-driven cooperative coevolution, in three simulated multirobot domains.

3.1 State of the Art

As discussed in Section 2.4.2, the issue of premature convergence in CCEAs is caused by the variability of the fitness evaluation — the fitness of an individual depends on the behaviour of the other coevolving team members, which can deceive the evolutionary process. One way of reducing premature convergence is therefore to reduce the variability of the evaluations. Previous works, discussed in Section 2.4.3, have shown that this can be achieved by evaluating the individuals together with *optimal* collaborators, or in a large number of collaborations. The objective is to assess the value of the individual with less variability and sensitivity to the other populations.

Existing studies with these evaluation schemes are, however, mostly focused on function optimisation domains (e.g. Panait et al., 2006b; Popovici and De Jong, 2005; Wiegand et al., 2001) and evolutionary game-theory (e.g. Panait, 2010; Wiegand and Potter, 2006; Wiegand et al., 2002), and always with only two coevolving populations. It is thus unclear whether the aforementioned methods for overcoming convergence to suboptimal equilibria are efficient and effective in the multirobot domains, for two main reasons:

- Existing approaches rely on the use of large numbers of collaborations to assess the fitness of each individual. When evolving controllers for robots, the number of generations can only be reduced to some extent — a large number of generations is typically needed for fine-tuning the controllers (which can have hundreds of parameters in the case of neural networks), even with a perfect fitness gradient. Every increase in the number of collaborators therefore results in a steep increase of computational complexity that is not viable in domains that rely on time-consuming simulations for evaluating the individuals.

- Existing approaches have only been demonstrated in coevolutionary systems with two populations, while multirobot systems are often composed of more than two agents. It is unclear how these approaches can be adapted to more populations, and what the consequences would be: the number of possible collaborations increases exponentially with the number of populations, so one might expect a steep increase in the number collaborations needed to evaluate each individual.

Analysing the previous works on the evolution of control for multirobot systems with CCEAs (e.g. Nitschke et al., 2012a; Potter et al., 2001; Yong and Miikkulainen, 2009, see Section 2.4.4 for more), the evaluation of each individual is usually conducted with a single collaboration, formed with the best individuals of each other population (the *single-best collaboration method*, Bucci and Pollack, 2002). To the best of our knowledge, the use of multiple collaborations has never been successfully demonstrated in a multirobot domain.

In a number of studies using multirobot domains, the issue of premature convergence is circumvented with the use of problem decomposition techniques (Panait and Luke, 2005a), such as incremental evolution (Gomez and Miikkulainen, 1997). Instead of directly addressing the problem of premature convergence, the given task is manually decomposed into a sequence of simpler sub-objectives, with the expectation that the evolutionary algorithm will incrementally be able to solve all sub-objectives. In an incremental evolution scheme, a series of evolutionary stages are defined by the experimenter, and evolution moves from one stage to the next when the population reaches a *sufficient* level of performance. In an environmental complexification setup, solutions are initially evaluated in a simplified version of the environment, which becomes progressively more complex as the evolutionary process is able to find solutions for the current stage. In another form of incremental evolution – staged evolution (Doncieux and Mouret, 2014) – the environment is the same throughout evolution, but the objectives change, with the fitness function rewarding simpler objectives earlier in evolution. Previous works have shown that incremental evolution schemes such as environmental complexification (Christensen and Dorigo, 2006; Gomez and Miikkulainen, 1997), can be successfully used to assist cooperative coevolution (Nitschke et al., 2012a,b; Yong and Miikkulainen, 2009). Uchibe and Asada, (2006) showed how staged evolution can also be used to integrate cooperative and competitive coevolution in a multirobot system. While the aforementioned decomposition techniques facilitate the evolution of complex behaviours, they require in-depth knowledge of the global task and how it can be divided into suitable sub-tasks (Doncieux and Mouret, 2014).

As discussed above, the existing approaches to avoid convergence to mediocre stable states either require a large number of evaluations, which is not feasible in multirobot domains, or require some form of task-specific knowledge/bias that has to be introduced by the experimenter. In this chapter, we study how diversity-oriented evolutionary techniques can be adapted to cooperative coevolutionary algorithms. Diversity-oriented techniques, such as novelty search (Lehman and Stanley, 2011a), see Section 2.5, have shown considerable success in tackling premature convergence issues in the field of evolutionary robotics, in a large number of studies (Doncieux and Mouret, 2014). Besides excelling at overcoming premature convergence, these techniques do not cause a significant increase in the evolutionary algorithm's computational complexity, and require little to none additional experimenter biases.

3.2 Novelty-driven Cooperative Coevolution

We propose and study three distinct approaches based on novelty search to overcome convergence to stable states in multi-population cooperative coevolution. The first approach, *NS-Team*, is based on traditional cooperative coevolution principles: an individual's novelty score is calculated based on the behaviour of the team in which it participated, without any discrimination of the individual agent behaviours. The second approach, *NS-Ind*, is based on the typical implementation of novelty search in non-coevolutionary algorithms: individuals are rewarded for exhibiting novel individual behaviours with respect to the other individuals in their population, thus maintaining behavioural diversity inside each population. The third approach, *NS-Mix*, is a combination of the first two: individuals are rewarded for displaying both novel individual behaviours and causing novel team behaviours.

3.2.1 Team-level Novelty

The team-level novelty approach (*NS-Team*) is described in Algorithm 2. In *NS-Team*, as in a typical cooperative coevolutionary algorithm (Algorithm 1), the evaluation of each individual begins with the formation of one team (a joint solution) composed of that individual and representative individuals from each of the other populations (step 8). The chosen representative of each population is the individual that obtained the highest team fitness score in the previous generation (step 14), or a random one in the first generation (step 4). The collective performance of the team is then assessed by evaluating it in the problem domain (step 9). *NS-Team* relies on the characterisation of the behaviour of a team as a whole. The novelty score of each individual is computed based on the team-level behaviour characterisation of the team with which it was evaluated. The novelty of the individual thus corresponds to the novelty of the team's behaviour. This process is analogous to the fitness assignment in typical CCEAs, in which an individual receives the fitness of the team in which it participated, without discriminating the individual's contribution.

Algorithm 2 *NS-Team*: Novelty-driven cooperative coevolution based on team-level behaviour characterisations.

1: Let $\mathcal{P}$ be the set of all populations in the coevolutionary system.
2: Let $\mathcal{A}$ be an archive of behaviour characterisations, initially empty.
3: **for each** population $p \in \mathcal{P}$ **do**
4: $\quad r_p \leftarrow$ randomly pick one individual $i \in p$
5: **for each** generation **do**
6: $\quad$ **for each** population $p \in \mathcal{P}$ **do**
7: $\quad\quad$ **for each** individual $i \in p$ **do**
8: $\quad\quad\quad t_i \leftarrow \{i\} \cup \{r_q : q \in \mathcal{P} \wedge q \neq p\}$ ▷ Form one team with the representatives
9: $\quad\quad\quad f_i, \mathcal{T}_i \leftarrow \texttt{Evaluate}(t_i)$ ▷ Obtain the team's fitness and behaviour characterisation
10: $\quad\quad$ **for each** $i \in p$ **do**
11: $\quad\quad\quad \eta_i \leftarrow \texttt{ComputeNovelty}(\mathcal{T}_i, \mathcal{A} \cup \{\mathcal{T}_x : x \in p\})$ ▷ Compute novelty based on the archive and the other individuals in the population
12: $\quad \mathcal{A} \leftarrow \texttt{UpdateArchive}(\mathcal{A}, \mathcal{T})$
13: $\quad$ **for each** $p \in \mathcal{P}$ **do**
14: $\quad\quad r_p \leftarrow$ individual $i \in p$ with maximum f_i
15: $\quad\quad p \leftarrow \texttt{Breed}(p)$ based on a combination of the scores f and η

Besides the team's novelty score (step 11), the team's fitness score is also taken into consideration in the selection and breeding process (step 15). The motivation for such combination is to drive evolution towards the exploration of *valuable* behaviour regions (as discussed in Section 2.5.5). The key difference is that while the team fitness measure is static, the team novelty measure is dynamic. Contrary to what happens in fitness-driven CCEAs, in *NS-Team* the attractors keep changing throughout evolution: what is novel in one generation will only remain so for a few generations. The evolutionary process is constantly led towards novel regions of the team behaviour space, which can avoid premature convergence to a single region of the solution space.

It should be noted that the method used to compute of the novelty scores (`ComputeNovelty`), the implementation of the archive update step (`UpdateArchive`), and the technique used to combine novelty and fitness (step 15), are independent of the *NS-Team* approach. We implemented these operations according to the results we obtained in a comprehensive empirical study (Gomes et al., 2015c), which are also consistent with common practices in novelty search studies, see Sections 2.5:

`ComputeNovelty`**:** The novelty score is computed based on the k nearest individuals in behaviour space, considering both the current population as well as an archive of past individuals.

`UpdateArchive`**:** The archive is updated every generation with randomly chosen individuals from the current population. The archive size is bounded for computational and memory efficiency. After the limit has been reached, randomly chosen individuals are removed to allow space for new ones.

`Breed`**:** The combination of novelty and team fitness objectives is achieved with Pareto-based multiobjective optimisation, following the NSGA-II algorithm (Deb, 2001). It should be noted that the proposed *NS-Team* algorithm only modifies the evaluation phase of the evolutionary algorithm, and therefore any underlying evolutionary algorithm can theoretically be used.

Team behaviour can be characterised using the design principles proposed in (Gomes et al., 2013): the behaviour characterisation focuses on the team as a whole, without directly discriminating between the respective contributions of individual agents. Such team-level characterisations can be crafted with task-specific knowledge (Gomes et al., 2013) or without it (Gomes and Christensen, 2013), as discussed in Section 2.5.4. Task-specific team-level characterisations can be based on measures of how the team influences the task environment, or by averaging agent's behavioural traits over all the members of the team. In this chapter, we use only task-specific characterisations, as it is the most clear approach, easier to analyse, and it is by far the most common in novelty search studies (Gomes et al., 2014e).

3.2.2 Individual-level Novelty

In domains where a high degree of cooperation is required for a joint solution to be successful, it may not be possible to assess the contribution of each agent to the success of the team. This issue is commonly known as the *credit assignment problem* (Colby and Tumer, 2015b; Potter and De Jong, 2000). Nonetheless, it is possible to describe the behaviour of each individual agent when participating in a team, ignoring to some extent whether the agent's actions are harmful or beneficial with respect to the team's objectives.

We additionally study a novelty-based coevolutionary algorithm that uses individual agent behaviour characterisations, instead of the team-level behaviour characterisations used in *NS-Team*. In *NS-Ind*, individuals are rewarded for displaying novel agent behaviours, regardless of the behaviour of the teams in which the individuals were evaluated. The objective of *NS-Ind* is to directly promote behavioural diversity inside each population, thus preventing premature convergence of the evolutionary process, following the previous successes of novelty-based techniques in single-population evolutionary algorithms (Doncieux and Mouret, 2014; Gomes et al., 2013; Lehman and Stanley, 2011a; Mouret and Doncieux, 2012).

The implementation of *NS-Ind* is detailed in Algorithm 3. The algorithm is similar to the novelty search implementation in non-coevolutionary algorithms, with one novelty archive (A_p) for each population p, and the novelty scores are computed only within each population. During the evaluation of an individual, the behaviour of that individual in the context of a team is characterised (step 9). This characterisation is then used to compute the novelty of the individual, by comparing it with the other behaviours observed in the respective population, and in the archive of that population (step 11). Since the computation of novelty for an agent is based exclusively on behaviours observed within its population, there can be a different behaviour characterisation functions (with different behaviour features) for each of the agents. In the experiments presented in this thesis, however, all agents are characterised using the same function.

Algorithm 3 *NS-Ind*: Novelty-driven cooperative coevolution based on agent-level behaviour characterisations.

1: Let $\mathcal{P}$ be the set of n populations in the coevolutionary system.
2: Let $(\mathcal{A}_1, \cdots, \mathcal{A}_n)$ be a list of archives of behaviour characterisations, all initially empty.
3: **for each** population $p \in \mathcal{P}$ **do**
4: $\quad r_p \leftarrow$ randomly pick one individual $i \in p$
5: **for each** generation **do**
6: $\quad$ **for each** population $p \in \mathcal{P}$ **do**
7: $\quad\quad$ **for each** individual $i \in p$ **do**
8: $\quad\quad\quad t_i \leftarrow \{i\} \cup \{r_q : q \in \mathcal{P} \wedge q \neq p\}$ ▷ Form one team with the representatives
9: $\quad\quad\quad f_i, \mathcal{I}_i \leftarrow \texttt{Evaluate}(t_i)$ ▷ Obtain the team fitness and the individual behaviour characterisation of i
10: $\quad\quad$ **for each** $i \in p$ **do**
11: $\quad\quad\quad \eta_i \leftarrow \texttt{ComputeNovelty}(\mathcal{I}_i, \mathcal{A}_p \cup \{\mathcal{I}_x : x \in p\})$ ▷ Compute novelty based on the corresponding archive and the other individuals in p
12: $\quad\quad \mathcal{A}_p \leftarrow \texttt{UpdateArchive}(\mathcal{A}_p, \{\mathcal{I}_x : x \in p\})$
13: $\quad$ **for** $p \in \mathcal{P}$ **do**
14: $\quad\quad r_p \leftarrow$ individual $i \in p$ with maximum f_i
15: $\quad\quad p \leftarrow \texttt{Breed}(p)$ based on a combination of the scores f and η

Finally, the novelty of the individual is combined with the fitness of the team in which it participated (step 15), in order to drive the coevolutionary system towards novel, high-quality solutions. As in *NS-Team*, in our experiments we use a multiobjective algorithm to combine the individual novelty and team fitness scores.

3.2.3 Mixed Novelty

We additionally propose and evaluate *NS-Mix*, which combines *NS-Team* and *NS-Ind*. In *NS-Mix*, the individuals are rewarded both for causing novel team behaviours *and* novel agent behaviours. The implementation relies on *NS-Team* and *NS-Ind*. The team-level novelty scores (η) are calculated according to Algorithm 2, while the individual-level novelty scores (η') are calculated according to Algorithm 3. The combined algorithm is described in Algorithm 4. These two sets of novelty scores, together with the team fitness scores, are used to select and breed the individuals of each population (step 17). In the experiments described in this chapter, we implement *NS-Mix* with a multiobjective algorithm, maximising the three scores: team novelty, individual novelty, and team fitness score.

Algorithm 4 *NS-Mix*: Novelty-driven cooperative coevolution based on both agent-level behaviour characterisations and team-level behaviour characterisations.

1: Let $\mathcal{P}$ be the set of n populations in the coevolutionary system.
2: Let $(\mathcal{A}_T, \mathcal{A}_1, \cdots, \mathcal{A}_n)$ be a list of archives of behaviour characterisations, all initially empty.
3: **for each** population $p \in \mathcal{P}$ **do**
4: $\quad r_p \leftarrow$ randomly pick one individual $i \in p$
5: **for each** generation **do**
6: $\quad$ **for each** population $p \in \mathcal{P}$ **do**
7: $\quad\quad$ **for each** individual $i \in p$ **do**
8: $\quad\quad\quad t_i \leftarrow \{i\} \cup \{r_q : q \in \mathcal{P} \wedge q \neq p\}$ ▷ Form one team with the representatives
9: $\quad\quad\quad f_i, \mathcal{T}_i, \mathcal{I}_i \leftarrow \texttt{Evaluate}(t_i)$ ▷ Obtain the team fitness, team behaviour characterisation, and the individual characterisation of i
10: $\quad\quad$ **for each** $i \in p$ **do**
11: $\quad\quad\quad \eta_i \leftarrow \texttt{ComputeNovelty}(\mathcal{T}_i, \mathcal{A}_T \cup \{\mathcal{T}_x : x \in p\})$ ▷ Compute novelty based on the team archive ($\mathcal{A}_T$) and the other individuals in p
12: $\quad\quad\quad \eta'_i \leftarrow \texttt{ComputeNovelty}(\mathcal{I}_i, \mathcal{A}_p \cup \{\mathcal{I}_x : x \in p\})$ ▷ Compute novelty based on the respective archive ($\mathcal{A}_p$) and the other individuals in p
13: $\quad\quad$ $\mathcal{A}_p \leftarrow \texttt{UpdateArchive}(\mathcal{A}_p, \{\mathcal{I}_x : x \in p\})$
14: $\quad$ $\mathcal{A}_t \leftarrow \texttt{UpdateArchive}(\mathcal{A}_T, \mathcal{T})$
15: $\quad$ **for** $p \in \mathcal{P}$ **do**
16: $\quad\quad$ $r_p \leftarrow$ individual $i \in p$ with maximum f_i
17: $\quad\quad$ $p \leftarrow \texttt{Breed}(p)$ based on a combination of the scores f, η and η'

3.3 Behaviour Exploration Analysis

When studying novelty-driven algorithms and premature convergence, it is extremely valuable to be able to analyse how the behaviour space is explored by the evolutionary process — it can reveal where the evolutionary process is focusing, and allows the identification of local optima (Gomes et al., 2013; Lehman and Stanley, 2011a). In this section, we describe a set of tools for analysing and visualising the progress of the evolutionary algorithm and the behaviour space exploration, which are used throughout this thesis.

3.3.1 Behaviour Exploration Metrics

In coevolutionary algorithms, previous works have focused on the analysis of the best individuals evolved in each population, at every generation (*best-of-generation* individuals) (Popovici and De Jong, 2006). By analysing the trajectory of such individuals over the evolutionary run, it is possible to visualise to which regions of the solution space the coevolutionary process is converging. However, since we study problems with more than two populations, and since individuals have multidimensional genomes/behaviours, the previously proposed methods cannot be directly applied to our domain. Instead, we rely on the behaviour of the teams that obtained the highest fitness score in a given generation (Definition 7).

Definition 7. Best-of-Generation (BoG) teams: The set of teams that obtained the highest fitness scores in each generation of a given evolutionary run.

Analysing the behaviour space exploration, based on all the evolved individuals, can also be an important tool to uncover the underlying evolutionary dynamics, especially in novelty-driven evolutionary approaches. For instance, Gomes et al., (2013) and Lehman and Stanley, (2011b) analyse the exploration of the behaviour space to discover the diversity of solutions evolved for a given task. In a novelty-based evolutionary process, looking only at the best-of-generation individuals can be misleading — since the evolutionary process is not exclusively driven by fitness, a large amount of lower fitness (but behaviourally diverse) solutions can be evolved.

We implemented three measures of exploration, to cover both the team behaviour space (*BoG team dispersion* and *All team dispersion*) and the individual agent behaviour space (*Individual dispersion*). For all three metrics, dispersion is given by the mean absolute difference among the behaviour characterisation vectors. Considering a set of behaviour characterisations $\varphi = \{\varphi_1, \cdots, \varphi_n\}$, the mean difference (MD) is given by:

$$\mathrm{MD}(\varphi) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{dist}(\varphi_i, \varphi_j)}{n(n-1)}\ , \tag{3.1}$$

where $dist$ is the Euclidean distance between the respective behaviour characterisation vectors. The mean difference (MD) is a non-parametric measure of statistical dispersion, that is not defined in terms of a measure of central tendency (Yitzhaki et al., 2003). A low mean difference value indicates that most teams/individuals have very similar behaviour characterisations, while a high value means the teams/individuals are well dispersed in the behaviour space. We defined the following metrics based on the MD:

Definition 8. BoG team dispersion: The behavioural dispersion of the best-of-generation teams evolved during a given evolutionary run. The *BoG team dispersion* is given by $\mathrm{MD}(\varphi')$, where φ' is the set composed of the respective team behaviour characterisations of all best-of-generation teams (Definition 7). This metric is intrinsically related to convergence — a low value means that the highest scoring teams always displayed very similar team behaviours, which suggests that evolution converged to a specific region of the team behaviour space.

Definition 9. All team dispersion: Similar to BoG team dispersion (Definition 8), but considering all teams evaluated during a given evolutionary run. *All team dispersion* is thus given by $\mathrm{MD}(\varphi)$, where φ is the set composed of the respective team behaviour characterisations of all the teams evolved during an evolutionary run.

Definition 10. Individual dispersion: The mean dispersion of the individual (agent) behaviours evolved in each population, averaged over all the populations. Considering $\mathcal{P}$ as the set of populations in the coevolutionary system, and φ_p the set of individual behaviour characterisations of all individuals evolved by population p, individual dispersion is given by:

$$\mathrm{ID}(\varphi) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathrm{MD}(\varphi_p) \tag{3.2}$$

3.3.2 Visualisation of the Best-of-Generation Teams

To be able to visualise the behaviour of the best-of-generation teams (similarly to Popovici and De Jong, 2006), the multidimensional behaviour space is reduced to two dimensions with Sammon mapping (Sammon, 1969). Sammon mapping (also known as Sammon projection) is an algorithm that maps a high-dimensional space to a space of lower dimensionality while trying to preserve the inter-point distances in high-dimensional space in the lower-dimension projection. Each point in the plot thus corresponds to one evolved team, where the distance between the points is related to the behaviour distance between the corresponding teams. The following procedure is followed to visualise the best-of-generation teams:

1. Gather the team behaviour characterisations of the best-of-generation teams, from all evolutionary runs under comparison.
2. Run the Sammon mapping algorithm with that entire set of characterisations, mapping each characterisation to a point in the continuous 2D space.
3. Individually plot the characterisations of each evolutionary run.

3.3.3 Behaviour Space Visualisation

To visualise the exploration of the behaviour space, i.e., the behaviours evolved by all individuals during an evolutionary run, we resort to Kohonen self-organising maps (Kohonen, 1990), as proposed in (Gomes et al., 2013). A Kohonen map is a two-dimensional discretised representation of a n-dimensional input space, which preserves the topological relations of the input space (i.e., vectors close in the input space are mapped to nearby positions in the two-dimensional map). The objective of using this approach for dimensionality reduction is the same as the Sammon mapping used to map the best-of-generation trajectory (Section 3.3.2). Sammon mapping, however, has a high computational complexity and becomes infeasible with a high number of input samples, while Kohonen maps scale well with the input size. We therefore use Kohonen maps to project the exploration of the behaviour space by all evolved individuals into two dimensions.

To visualise the exploration of the behaviour space in the different evolutionary runs, or by different methods, we follow this methodology:

1. The Kohonen map is trained with a random sample of all the behaviours found, in all evolutionary runs of all methods under comparison.
2. The behaviours evolved in each evolutionary run are separately mapped to the trained map: each individual is assigned to the node (behaviour space region) with the closest weight vector.

3. We count and plot how many individuals were assigned to each node, thus obtaining the dispersion of the evolved individuals over the behaviour space.

3.4 Evaluation in the Predator-prey Task

Predator-prey pursuit is one of the most common domains studied in multiagent coevolution, both in cooperative coevolution (e.g., Nitschke et al., 2012b; Yong and Miikkulainen, 2009, see Section 2.4.4 for more) as well as in competitive coevolution (Nolfi, 2012; Rawal et al., 2010). Predator-prey tasks involve a number of agents (predators) chasing a prey. The predators cannot move faster than the prey, and they therefore need to cooperate in order to successfully capture the prey. In cooperative coevolution studies, only the team of predators is evolved, while the prey has a prespecified fixed behaviour. The predator-prey task is especially interesting in cooperative coevolution studies because heterogeneity in the predator team is required to effectively catch the prey, along with a tight coordination among the predator (Yong and Miikkulainen, 2009). In this section, we describe the predator-prey domain used in our study, and the experiments conducted with this domain.

3.4.1 Predator-prey Task

The predators are initially placed in linear formation at one end of the arena, in the slots depicted in Figure 3.1a. We defined task variants with the number of predators ranging from two to seven. A single prey is randomly placed near the centre of the arena. The arena is not physically bounded, and if the prey escapes the arena, the trial ends. The task parameters are listed in Appendix A.2. We use a version of the task where the predators cannot communicate nor sense one another (Rawal et al., 2010; Yong and Miikkulainen, 2009). Each predator is controlled by a neural network that receives only two inputs: (i) the distance to the prey (D_p), and (ii) the relative orientation of the agent with respect to the prey (α_p). These inputs are normalised before being fed to the neural network, and the network's two outputs control respectively the speed (D_m) and the rotation (α_m) of the agent, see Figure 3.1b. The neural network that controls each predator is a fixed-topology recurrent Jordan network (Jordan, 1997), see Figure 3.1c.

The predators move at most at the same speed of the prey (1 unit/step). The behaviour of the prey consists of moving away from any predator within a radius of V around the prey. If there are no predators within that radius, the prey does not move. Otherwise, the prey moves at a constant speed, with a direction opposite to the centre of mass of the nearby predators. We use task variants with different values for the radius V, ranging from 4 to 13 units. The prey is captured if a predator collides with it. A trial ends if the prey is captured, escapes the arena, or if T=300 simulation steps elapse. Each team of predators is evaluated in five simulation runs, varying the starting position of the prey.

The fitness function F_{pp} is based on previous works (Nitschke et al., 2012b; Yong and Miikkulainen, 2009). If the prey was captured, F_{pp} increases as the time to capture the prey (τ) decreases, otherwise it increases as the average final distance from the predators to the prey (d_f) decreases:

$$F_{pp} = \begin{cases} 2 - \tau/T & \text{if prey captured} \\ max(0, (d_i - d_f)/size) & \text{otherwise} \end{cases}, \tag{3.3}$$

where T is the maximum simulation length, d_i is the average initial distance from the predators to the prey, and $size$ is the side length of the arena.

The behaviour characterisations were defined based on systematically derived behaviour characterisations (SDBC) (Gomes et al., 2014e, see Section 2.5.4). We chose a subset of the extracted features, based on the estimated relevance of the features with the predator-prey task. The team-level behaviour characterisation $\beta_{pp}(t)$ is a vector of length 4. The agent-level characterisation of an agent $\beta_{pp}(a)$ is based on behaviour features similar to $\beta_{pp}(t)$, but measured for a specific agent instead of the whole team. The characterisations are described in Table 3.1.

TABLE 3.1: Behaviour characterisations used in the predator-prey task. All features have values normalised to the range [0,1].

Team-level characterisation $\beta_{pp}(t)$	Individual-level characterisation $\beta_{pp}(a)$
▷ Whether the prey was captured	▷ Whether agent a captured the prey
▷ Average final distance of the predators to the prey	▷ Final distance of a to the prey
▷ Average distance of each predator to the other predators over the simulation	▷ Average distance of a to the other predators over the simulation
▷ Simulation length	

3.4.2 Evolutionary Setup

We use a canonical genetic algorithm to evolve the neural networks that control the agents: the weights of the networks are directly encoded in the chromosomes; the algorithm uses tournament selection; the genes (weights) are mutated individually with a fixed probability; we apply one-point crossover; and the elite of each population passes directly on to the next generation. Novelty-driven cooperative coevolution is implemented as described in Section 3.2, and configured according to

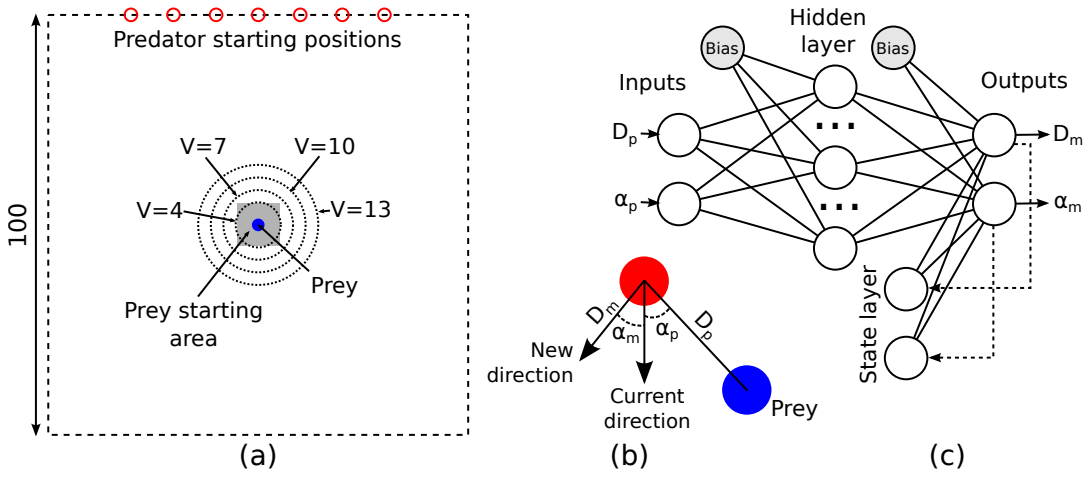


FIGURE 3.1: Predator-prey task setup. (a) Initial conditions of the simulation, with the possible prey vision ranges (V) and the possible predators' starting positions (circles at the top). (b) Sensors and effectors of each predator: the predator senses the distance D_p and relative orientation α_p of the prey, and the effectors control the speed D_m and turning angle α_m. (c) The structure of the neural network controller of each predator.

the results presented in (Gomes et al., 2015c): the nearest neighbours among the current population and the archive are used for novelty computation; and the archive is composed of randomly chosen individuals. See Appendix A.1 for parameter values.

Each experimental treatment was repeated in 30 evolutionary runs. In all experiments, the highest scoring team of each generation was re-evaluated *a posteriori* in 50 simulation trials. As the initial position of the prey is stochastic, the re-evaluation yields a more accurate estimate of the team fitness. All the team fitness plots presented in the chapter are based on the scores obtained in this post-evaluation.

3.4.3 Base Fitness-driven Cooperative Coevolution

In the first set of experiments, we analyse how fitness-based coevolution performs when faced with varying degrees of task difficulty. We vary the difficulty of the task by varying the prey's visual range (V). Increasing V allows the prey more room and time to escape from the predators. As such, a higher degree of cooperation, as well as a more fine-tuned strategy, are required in the team of predators in order to successfully catch the prey. In setups with high V values (*V10*, *V13*), only one non-cooperating agent might be sufficient to compromise the performance of the whole team, as it can drive the prey away or leave room for it to escape. Figure 3.2 shows performance achieved with fitness-driven coevolution for three predators and with values of V varying from 4 to 13.

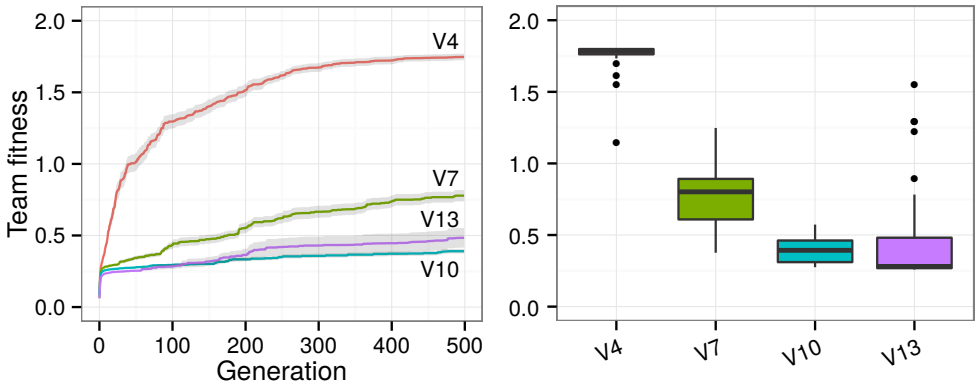


FIGURE 3.2: Team fitness scores achieved with fitness-based evolution in task setups with varying prey vision (V). Left: highest fitness scores achieved so far at each generation, averaged over 30 runs for each setup. The grey areas depict the standard error. Right: boxplots of the highest scores achieved in each evolutionary run, for each task setup. The whiskers represent the highest and lowest value within 1.5 IQR, and the dots indicate outliers.

The results show that fitness-driven coevolution (*Fit*) is only able to consistently evolve effective solutions in the easiest setup ($V4$). In the other setups, *Fit* rarely reaches high-quality solutions. It should be noted that it is possible to find effective solutions for all these task setups, as it will be shown in the following sections. To determine the reason for failure, we analyse the best-of-generation (BoG) teams, as described in Section 3.3.1. Figure 3.3 shows the behaviour of the BoG teams in representative evolutionary runs[1], along with the mean value of the BoG team dispersion (D) for each task setup. These results show that in the easier task setup ($V4$), coevolution can consistently explore the behaviour space and reach regions of the behaviour space where high-quality solutions can be found. In the other task setups,

[1]For each setup, we chose the evolutionary run that had a value of *BoG dispersion* (D) closest to the mean of all runs in that setup.

however, coevolution converges prematurely to a narrow region of the behaviour space, resulting in a relatively low degree of BoG team dispersion (D).

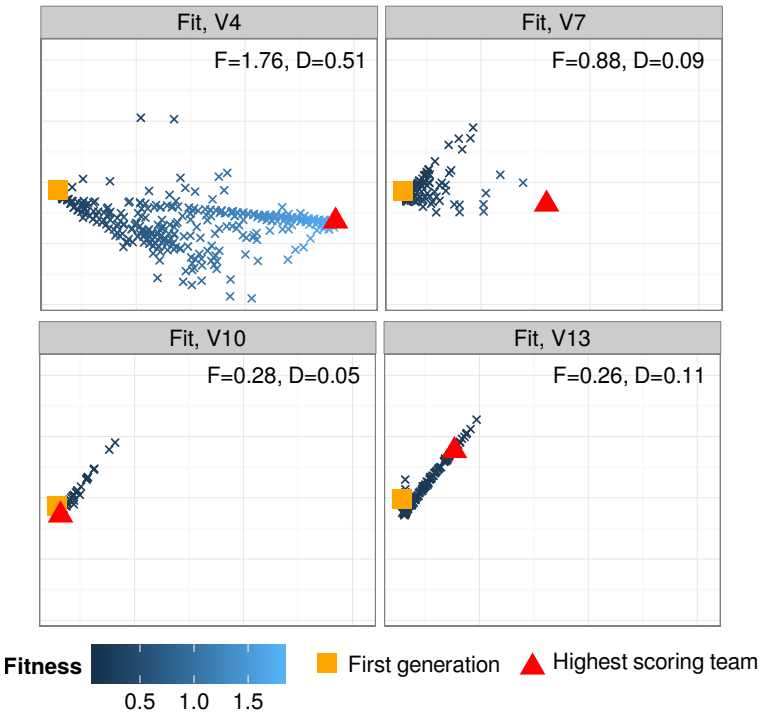


FIGURE 3.3: Behaviour of the best-of-generation teams in representative evolutionary runs of fitness-driven coevolution. Each cross represents one team, mapped according to its team behaviour. The four-dimensional behaviour space was reduced to two dimensions using Sammon mapping, see Section 3.3.1. D is the BoG team dispersion of the respective run, and F is the highest fitness score achieved.

3.4.4 Increasing the Number of Collaborations

We experimented with techniques studied in previous works to try to overcome convergence to suboptimal solutions in fitness-driven coevolution. As suggested by Wiegand et al., (2001), we increased the number of random collaborations with which each individual is evaluated. The fitness assigned to an individual is the maximum reward it obtained with any collaboration. To evaluate each individual, $(N + 1)$ collaborations are formed: one with the best individuals from the previous generation, and N collaborations composed with randomly chosen collaborators. According to previous results (Panait, 2010; Popovici and De Jong, 2005; Wiegand et al., 2001), increasing the number N of collaborations should increase the likelihood of the coevolutionary algorithm to converge to the global optimum. We therefore evaluated how varying the number N impacts the performance of fitness-based coevolution. Since this scheme has, to the best of our knowledge, only been used in coevolutionary setups with two populations, we also experimented with a task setup with only two predators (*V4/2*) to establish a fair basis for comparison. In the remaining setups, three predators are used, and the random collaborations are formed by partnering the individual that is being evaluated with one randomly chosen individual from each of the other populations.

Figure 3.4 (left) shows the effect of increasing N in the different task setups. It should be noted that the number of generations was the same (500) in all evolutionary configurations, meaning that the number of evaluations increases linearly

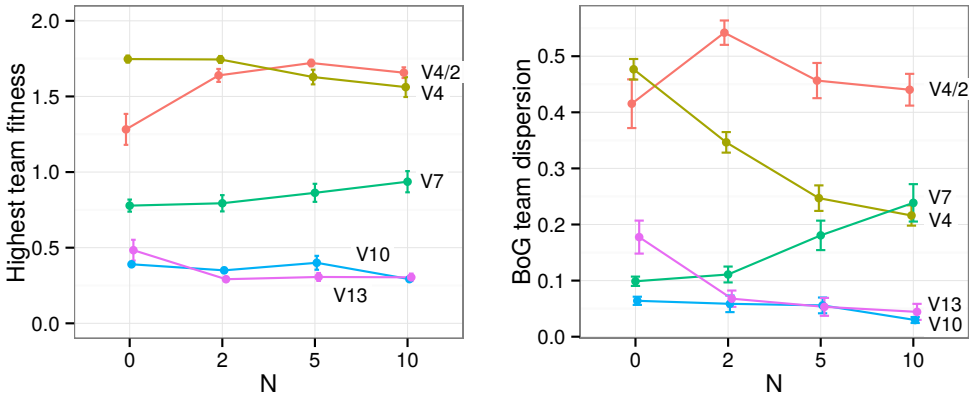


FIGURE 3.4: Left: highest team fitness scores achieved in each evolutionary run, for each task setup with varying task difficulty (prey's vision range – V), and a varying number of random collaborators (N). The *V4/2* setup uses only two predators, while the other setups use three predators. Right: behavioural dispersion of the best-of-generation (BoG) teams. Standard error bars are shown.

with N. The results show that using random collaborations can significantly improve the performance of fitness-based coevolution in the two-population setup (*V4/2*, $p = 0.005$, Kruskal-Wallis test), with respect to the highest team fitness scores achieved. This result is coherent with previous studies performed with two-population setups. The results obtained in the three-agent setups, however, reveal a substantially different trend. In *V7*, no significant differences in the performance were found ($p = 0.389$, Kruskal-Wallis), and in *V4*, *V10* and *V13*, increasing the number of collaborations can actually result in a lower performance ($p = 0.019$, $p < 0.001$, and $p = 0.006$ respectively).

Figure 3.4 (right) shows the influence of N on behavioural convergence (as defined in Section 3.3.1). In the setups *V4*, *V10*, and *V13*, increasing the number of collaborations led to an increase in convergence to specific region of the solution space ($p < 0.001$, Kruskal-Wallis test), which in turn correlates with inferior performance. In the other setups, the influence of N is less clear.

Our results suggest that the traditional methods of overcoming convergence to stable states may not be effective in coevolutionary setups with more than two populations, and with a large number of individuals. Panait, (2010) demonstrated that a CCEA converges to the global optimum if a *sufficient* number of collaborations are used to evaluate each individual. An insufficient number of random collaborations might lead to poor fitness estimates that can result in convergence to suboptimal solutions. A *sufficient number of random collaborations* is, however, highly problem-dependent (Panait, 2010). As the number of possible collaborations increases exponentially with the number of populations and the number of individuals, the collaborations required to obtain a proper estimate of an individual's fitness may also increase significantly, maybe even exponentially. Unfortunately, our results do not provide a definite answer to this question, as exponentially increasing the number of collaborations is typically not computationally feasible in the domain of embodied multiagent systems.

3.4.5 Novelty-driven Coevolution

In this section, we analyse how novelty-driven cooperative coevolution can overcome the problem of premature convergence. We compare fitness-driven coevolution (*Fit*) with team-level novelty (*NS-Team*), individual-level novelty (*NS-Ind*), and

a combination of the two (*NS-Mix*). In the novelty-based approaches, a multiobjective algorithm, NSGA-II (Deb et al., 2002), is employed to combine the novelty and fitness objectives, as described in Section 3.2. Based on the previously discussed results, we did not use random collaborations in any of the experimental setups: each individual is evaluated together with the individuals of the other populations that obtained the highest fitness scores in the previous generation. Three predators were used in all experiments.

Overcoming premature convergence

Figure 3.5 (left) shows the highest team fitness scores achieved with each method, for each level of task difficulty. Figure 3.5 (right) shows the behavioural dispersion of the best-of-generation teams, which, as discussed above, can be a good indicator of premature convergence. Figure 3.6 shows the highest fitness score achieved at each generation, averaged over the 30 evolutionary runs.

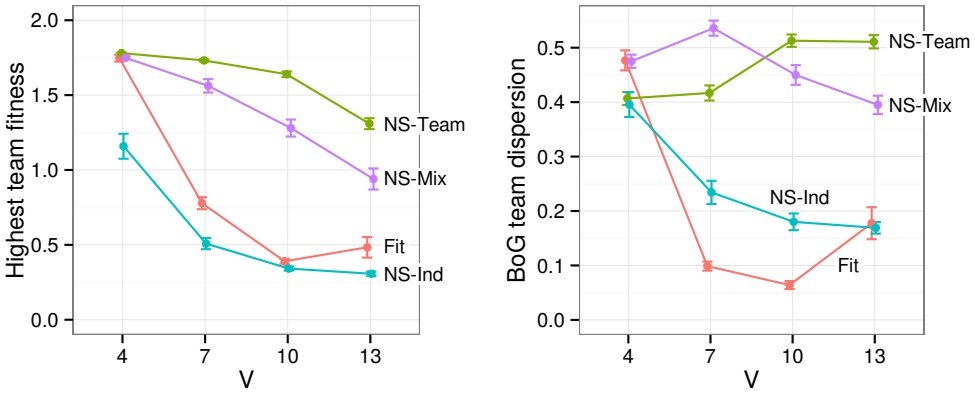


FIGURE 3.5: Left: Highest team fitness scores achieved in each evolutionary run with the different methods, for each task setup with varying task difficulty (V). Right: Behavioural dispersion of the best-of-generation teams. Standard error bars are shown.

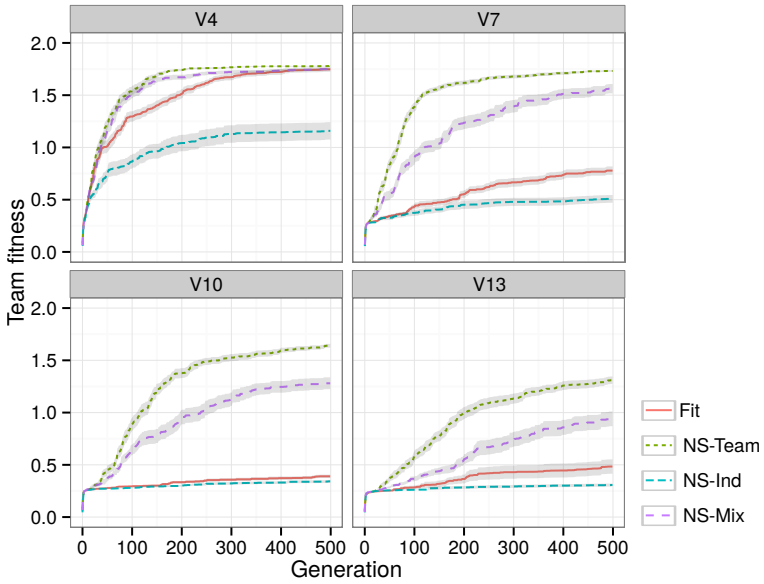


FIGURE 3.6: Performance of fitness-based evolution and the novelty-based approaches in each task setup. The plots show the highest team fitness scores achieved so far at each generation, averaged over 30 runs for each method. The grey areas depict the standard error.

As discussed in the previous section, the performance of fitness-driven coevolution drastically decreases as the difficulty of the task is increased. *NS-Team*, on the other hand, can consistently evolve effective solutions (fitness > 1.0) in all task setups. The average team fitness of the best solutions evolved by *NS-Team* is significantly superior to the solutions evolved by *Fit* in all setups except the easiest one (adjusted[2] $p < 0.001$, Mann-Whitney test). *NS-Team* was clearly the highest performing approach among the novelty variants, while *NS-Ind* displayed the lowest performance. *NS-Ind* could not consistently evolve effective solutions, even in the easiest task setup, and was significantly inferior to all other novelty-based methods (adjusted $p < 0.001$). The performance of *NS-Mix* was significantly superior to *NS-Ind* in all setups, but it was inferior to *NS-Team* (adjusted $p < 0.01$). *NS-Team* could also achieve higher quality solutions earlier in the evolutionary process, see Figure 3.6.

As the results in Figure 3.5 (right) show, both *NS-Team* and *NS-Mix* were able to overcome the issue of premature convergence to stable states. In Figure 3.7, we show the dispersion patterns of the best-of-generation teams, for a representative evolutionary run of each setup. Although *NS-Team* can still get attracted to low-quality regions of the collaboration space (especially in *V10* and *V13*), it is ultimately capable of escaping these regions, and can reach high-quality collaborations. *NS-Ind*, on the other hand, was mostly ineffective. Rewarding novel individual behaviours was not an effective strategy to avoid premature convergence to narrow regions of the team behaviour space.

Exploration of the behaviour space

We resorted to the dispersion measures that use all teams (*all team dispersion*) and all individuals (*individual dispersion*), see Section 3.3.1, to better understand the difference between the proposed novelty search implementations. The results are shown in Figure 3.8. Fitness-driven coevolution always displays significantly inferior degrees of dispersion (adjusted $p < 0.001$, Mann-Whitney test) when compared to *NS-Team*, both in terms of the dispersion of team behaviours (Figure 3.8 left), and individual behaviours (Figure 3.8 right). These results confirm the attraction of fitness-based coevolution to stable states, already discussed above. Novelty-driven coevolution (*NS-Team*) displays substantially different evolutionary dynamics, and does not seem to converge to specific regions of the collaboration space. It explores a much wider range of collaborations (team behaviours), and can reach more collaboration regions associated with high-quality behaviours.

As previously mentioned, the performance of individual-level novelty (*NS-Ind*) was substantially inferior to *NS-Team*, failing to achieve effective solutions across all task setups. As the results in Figure 3.8 (right) show, *NS-Ind* is effective in discovering a reasonable diversity of agent behaviours, when compared to *NS-Team* and *Fit* (adjusted $p < 0.001$). However, this diversity of agent behaviours does not translate into the discovery of novel collaborations (Figure 3.8, left). The individual novelty objective is not aligned with the team novelty objective, despite the similarity in the dimensions of the individual-level and team-level behaviour characterisations (see Table 3.1). Cooperation is not directly taken into account in *NS-Ind*, which results in poorly performing joint solutions.

The set of team behaviours discovered by *NS-Ind* was the least diverse, among the considered novelty-based treatments ($p < 0.001$). To directly encourage some

[2]When multiple comparisons within the same set of results were made, the p-values were adjusted with the Holm-Bonferroni correction.

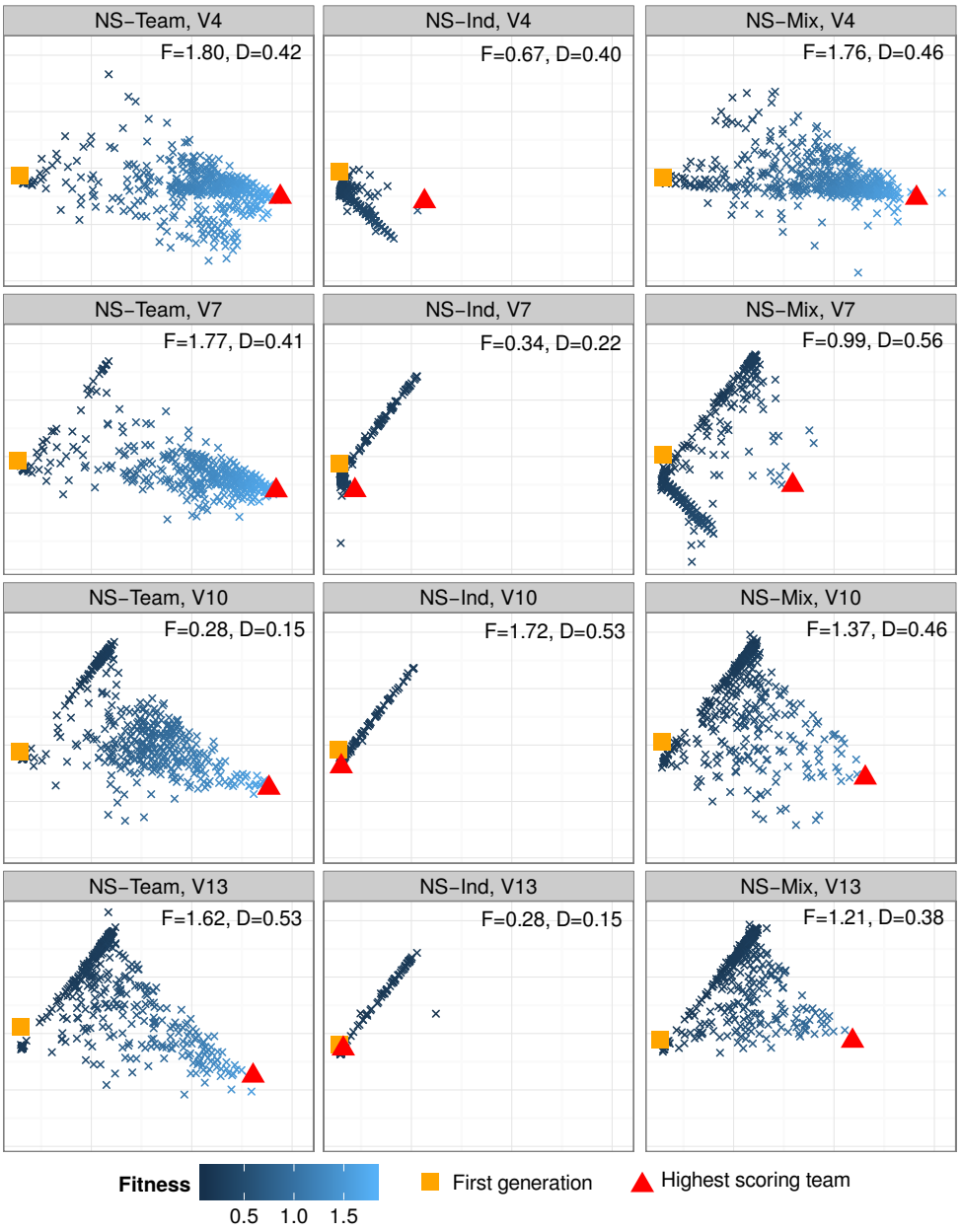


FIGURE 3.7: Behaviour of the best-of-generation teams in representative evolutionary runs. The behaviour space was reduced to a two-dimensional space with Sammon mapping.

degree of exploration of team behaviours, we also proposed *NS-Mix*: a combination of *NS-Team* and *NS-Ind* where the team novelty objective is added to the individual novelty and team fitness objectives. *NS-Mix* increased both the team and individual behavioural diversity when compared to *NS-Ind*. The diversity of team behaviours, however, was still significantly inferior to *NS-Team* ($p < 0.001$).

The results obtained with *NS-Ind* and *NS-Mix* suggest that favouring diversity of individual agent behaviours can actually be harmful. As each separate population is encouraged to constantly evolve towards individual behavioural novelty, it might be hard to form effective collaborations, as the individuals of a population do not have neither enough time nor incentive to adapt to the other populations. This evolutionary dynamic is contrary to what occurs in *NS-Team*, where each population can specialise in one area of the agent behaviour space at a time, thus allowing

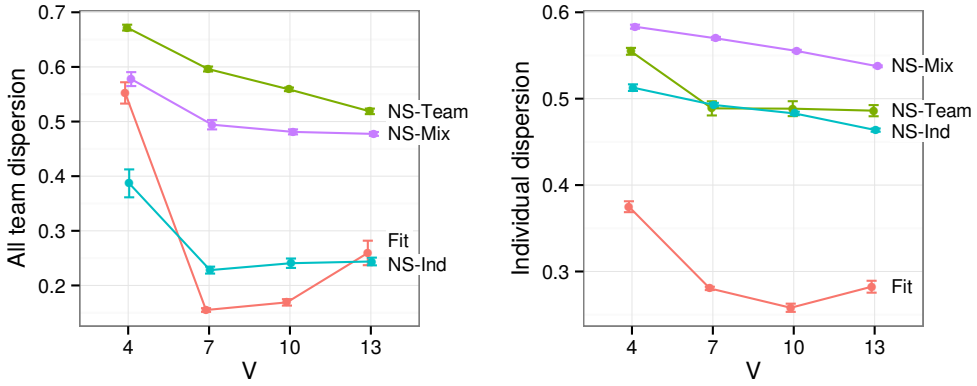


FIGURE 3.8: Analysis of team behaviour dispersion, considering all the evolved teams (left), and individual behaviour dispersion (right), with each evolutionary treatment, for task setups with varying difficulty (V).

a better adaptation of the populations to each other. Overall, we showed that for the purpose of achieving effective solutions, novelty search with team-level characterisations was the most effective method of introducing novelty search in cooperative coevolutionary algorithms.

3.4.6 Solution Diversity

Besides the ability of overcoming premature convergence, novelty search has been shown capable of discovering a wide range of solutions for a given task (Gomes et al., 2013; Lehman and Stanley, 2011b). In this section, we present a qualitative analysis of the exploration of the behaviour space and diversity of solutions evolved. To make a fair comparison between the diversity of solutions evolved, we used the task setup with three predators and $V = 4$, since this was the only setup where *Fit* and *NS-Team* achieved similar team fitness scores (Figure 3.5).

The four dimensions of the behaviour characterisation were reduced to two dimensions using a Kohonen self-organising map in order to obtain a visual representation of the team behaviour space exploration (see Section 3.3.1). The trained Kohonen map is depicted in Figure 3.9 (top), and in Figure 3.9 (bottom), we show the behaviour exploration in a typical evolutionary run of *Fit* and *NS-Team*.

As discussed in Section 3.4.5, fitness-driven coevolution often explores a relatively narrow region of the behaviour space (corresponding to the top-right corner of the map, Figure 3.9), and converges to solutions in regions (10,9) and (10,8) (Figure 3.9) in all evolutionary runs. *NS-Team* evolves individuals that cover a wider range of behaviour regions, and can find diverse high-quality solutions. These results are consistent with the exploration measures in Section 3.4.5. To confirm the diversity of solutions, we inspected the highest scoring solutions found in the high-quality behaviour regions. Figure 3.10 depicts typical movements of the predators and the prey in the different solutions. It is noteworthy that, for this task difficulty level ($V = 4$), *NS-Team* discovered solutions where only two predators actually chase the prey (see for instance regions (8,1) and (10,2)), which highlights the diversity of team behaviours that *NS-Team* can evolve.

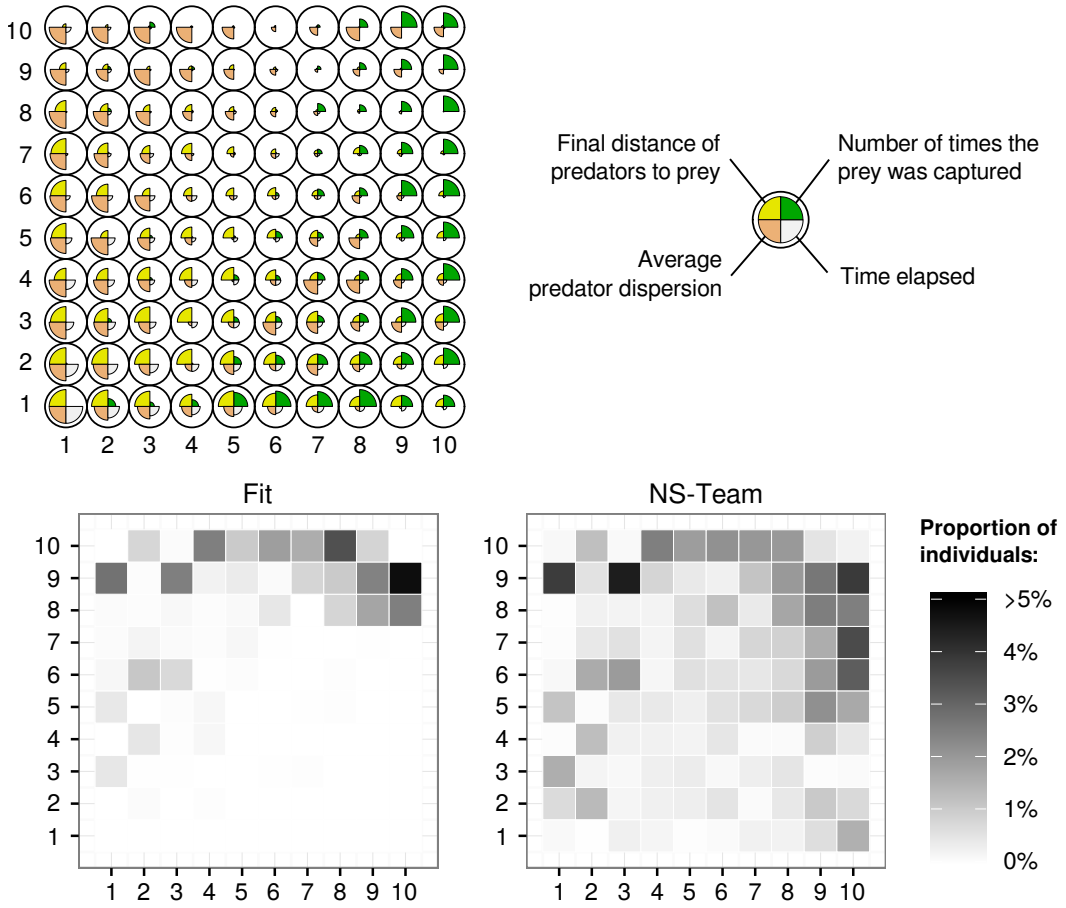


FIGURE 3.9: Top: trained Kohonen map, where each unit represents a region of the team behaviour space. The high-quality behaviour regions (where the prey is caught most of the times) are found along column 10 and near it. Bottom: team behaviour exploration in a typical evolutionary run of fitness-based coevolution and *NS-Team*, with the easiest task setup (*V4*). The darker a region, the more individuals were evolved belonging to that behaviour region.

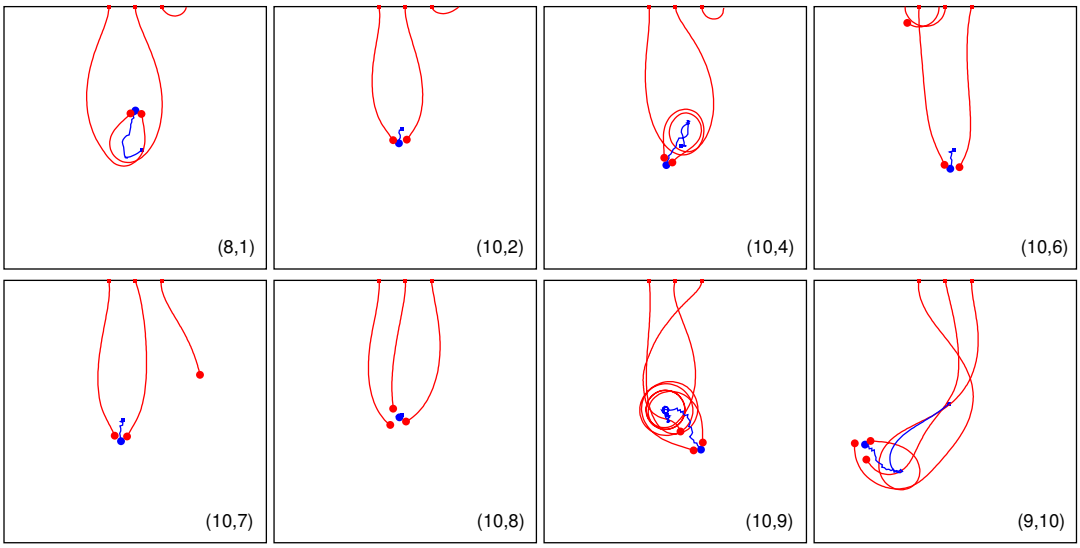


FIGURE 3.10: Examples of solutions evolved by *NS-Team* in the *V4* task setup, found in the behaviour regions associated with high-quality solutions (the numbers indicate the coordinates in the plots in Figure 3.9). The three preys (red) start at the top of the arena, and the prey (blue) starts in the centre.

3.4.7 Scalability with Respect to Team Size

We conducted evolutionary runs in setups with between two and seven predators to assess the scalability of *NS-Team* with respect to the number of populations, see Figure 3.11 (left). To assess if *NS-Team* is able to take advantage of the higher number of available agents, we analyse how many predators actually participate in catching the prey, compared to the total number of predators. We consider a predator as *participant* if it is near the prey (within $1.5 \times V$) in the moment the prey is caught, as the predators typically surround the prey in order to catch it (see for instance Figure 3.10). Figure 3.11 (right) shows the mean number of participant predators in each setup, considering the best-of-generation individuals only.

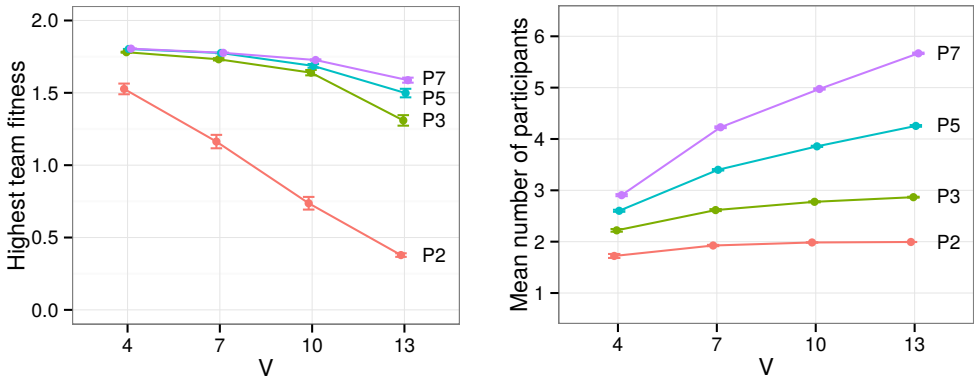


FIGURE 3.11: Left: Highest team fitness scores achieved with *NS-Team* in task setups with multiple combinations of number of predators and prey vision range V. Right: Mean number of participant predators in the best-of-generation solutions evolved in each setup.

As the results in Figure 3.11 (left) show, adding more predators to the system never negatively impacts the performance of *NS-Team*. In the most challenging setups, *V10* and *V13*, adding more agents always resulted in a significant improvement of the team fitness scores achieved by *NS-Team* ($p < 0.05$, Mann-Whitney). The results in Figure 3.11 (right) confirm that the highest scoring solutions take advantage of the higher number of predators available, even though a smaller number of agents is often enough to solve the task. For a given task difficulty level (V), adding more predators always resulted in a significantly higher number of participant predators ($p < 0.001$). Overall, our results suggest that *NS-Team* can scale with the number of populations – it performed well with up to seven populations, and was able to take advantage of all or most of the agents available.

3.4.8 Combination of Novelty and Team Fitness

In all the experiments described so far in this section, the novelty-based approaches always consisted of one or two novelty objectives combined with the team fitness objective through a multiobjective algorithm, as described in Section 3.2. This choice was based on previous findings that show that the combination of novelty and fitness is the most effective way of applying novelty search in optimisation problems (Gomes et al., 2015c; Lehman et al., 2013). Nevertheless, a number of previous works also show that, in some situations, novelty search alone might suffice to solve challenging tasks (Gomes et al., 2013; Lehman and Stanley, 2011a). In this case, the only drive in the evolutionary process is behavioural novelty and the quality of the evolved solutions is completely ignored.

In this section, we evaluate the necessity of combining novelty with team fitness. We only focus on *NS-Team*, since it is clearly the best performing approach. We introduce *NS*-Team*, which is implemented similarly to *NS-Team* (see Algorithm 2), with the following differences:

1. The selection score of each individual is simply the team novelty score that individual obtained – the behavioural novelty of the team with which the individual was evaluated.

2. The representative of each population is the individual that obtained the highest novelty score in the previous generation, or a random one in the first generation. We chose the most novel individual as the representative in order to avoid introducing any biases from the fitness function in the evolutionary process.

In Figure 3.12, we compare *NS-Team* with *NS*-Team*, and present fitness-driven coevolution as baseline. We use different task difficulty levels (V), and the number of predators is always three. The results show that the performance of pure novelty search (*NS*-Team*) is significantly inferior to the multiobjectivisation of novelty and team fitness (*NS-Team*) across all task setups ($p < 0.001$, Mann-Whitney). Nonetheless, it is noteworthy that the performance of *NS*-Team* was never significantly inferior to fitness-driven coevolution, and actually managed to achieve a significantly higher performance in the *V7* and *V10* task setups ($p < 0.001$). As novelty search encourages the exploration of behaviour regions that have not been visited so far, the coverage of the behaviour space is greater, and it is thus more likely to discover solutions in behaviour regions associated with high fitness scores. Our results thus suggest that novelty-driven coevolution can achieve high-quality cooperative solutions without explicitly looking for them in the first place, which is consistent with previous non-coevolutionary novelty search studies (Gomes et al., 2013; Lehman and Stanley, 2011a).

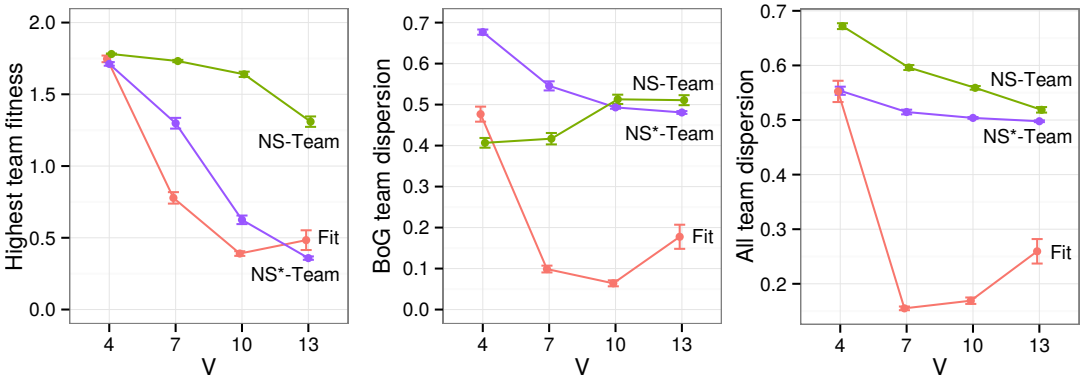


FIGURE 3.12: Comparison of pure novelty search (*NS*-Team*) and the multiobjectivisation of novelty and team fitness objectives (*NS-Team*). Left: highest team fitness scores achieved in each evolutionary run, with each approach and in each task setup. Middle: behavioural dispersion of the best-of-generation teams. Right: behavioural dispersion of all the evolved teams.

The results in Figure 3.12 (middle) show that pure novelty search is effective in avoiding convergence to stable states across all setups, and can find a good diversity of team behaviours (Figure 3.12, right). The lack of a team fitness objective, however, makes the behavioural exploration rather ineffective: in the more demanding task setups, pure novelty search fails to reach the high-quality regions of the collaboration space.

3.5 Validation with the Cooperative Foraging and Herding Tasks

We evaluate the proposed approaches in two additional robotics tasks, the cooperative foraging task and the herding task, to assess the general applicability of novelty-driven cooperative coevolution. These tasks require more complex neural controllers than the predator-prey task (Section 3.4), as the agents have a significantly higher number of sensors and effectors. To deal with this higher complexity, we use the NEAT algorithm (Stanley and Miikkulainen, 2002) to evolve the neural controllers for the agents.

3.5.1 Cooperative Foraging Task Setup

The cooperative foraging task requires a team of two agents to find and collect items in the environment. The two agents are simultaneously needed to collect each item, with each one using a different actuator (each agent can only use one actuator at a time). Behaviour specialisation and cooperation in the team is therefore required to solve the task. This task is inspired by collective foraging tasks commonly studied with both homogeneous (Bayındır, 2016) and heterogeneous (Nitschke et al., 2010) multiagent systems. This task is challenging because the agents must find each other in the environment, then they need to find the items, and complement each other to successfully collect them.

The task environment is depicted in Figure 3.13 (left), and the experimental parameters are listed in Appendix A.3. Eight items are placed randomly inside an arena bounded by walls. The two agents start in random locations and with random orientations. Each agent has the following sensors: (i) two short-range sensors ($c_{1,2}$) to detect collisions, (ii) three sensors for the detection of items ($r_{1..3}$), (iii) three sensors for the detection of the other agent ($d_{1..3}$), and (iv) one sensor that returns the type of the actuator currently used by the nearby agent. Two outputs control the linear speed and turning angle of the agent, and two other outputs determine which item collection actuator should be active (or none). When an item collection actuator is active, the agent remains still. To collect an item, the two agents need to be simultaneously over the item, with one agent having its *type 1* actuator active, and the other agent the *type 2* actuator. The item disappears from the environment when it is collected.

The fitness function F_r corresponds to the number of items collected during the simulation trial. The behaviour characterisations are listed in Table 3.2. Each team of individuals is evaluated in ten independent simulation trials.

3.5.2 Herding Task Setup

In the herding task (Potter et al., 2001), a group of shepherds must drive one or more sheeps into a corral. Additionally, foxes can also be present, which try to capture the sheep, and must be kept away by the shepherds. In our task setup, there are four shepherds, one sheep, and two foxes. As shown by Potter et al., (2001), the presence of foxes increases the number of skills required to solve the task, and as such behavioural specialisation within the shepherds group might be required to solve the task. Only the controllers for the shepherds are evolved. The shepherds are physically homogeneous.

The initial conditions of the herding task are depicted in Figure 3.13 (right). Each fox is placed randomly at the right side of the arena. The shepherds and the sheep

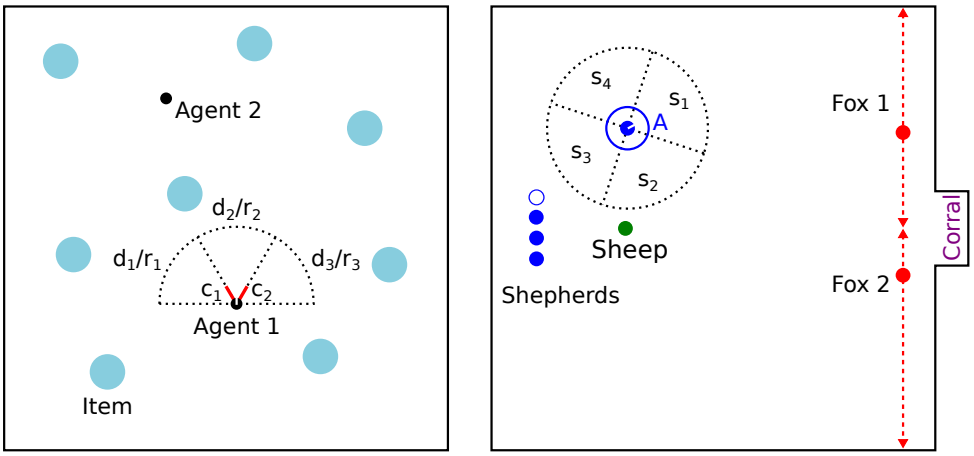


FIGURE 3.13: Left: an example of the initial conditions in the cooperative foraging task. Both agents have the same sensor and effector setup, although the setup is only shown for *Agent 1*. Right: initial conditions in the herding task. The shepherds start in a linear formation. In the figure on the right, we have moved the top-most shepherd to show the sensor setup. The two foxes are placed randomly along the respective line segment.

TABLE 3.2: Behaviour characterisations used in the cooperative foraging task. All means are taken over the simulation time, and all features are normalised to the range [0,1].

Team-level characterisation $\beta_r(t)$	Individual-level characterisation $\beta_r(a)$
▷ Mean distance of each agent to the nearest item, averaged over the 2 agents	▷ Mean distance between a and the nearest item
▷ Mean movement speed, averaged over the 2 agents	▷ Mean movement speed of a
▷ Mean distance between the agents	▷ How long the *type 1* actuator was activated by a
▷ Number of items collected	▷ How long the *type 2* actuator was activated by a

have fixed initial positions. Each shepherd has the following sensors: (i) four sensors that return the distance of the nearest shepherd ($s_{1..4}$); and (ii) eight sensors that return the distance and relative orientation of the sheep, the two foxes, and the centre of the corral. The two outputs control respectively the linear speed and turning angle of the shepherd.

When a shepherd approaches the sheep or one of the foxes (distance inferior to the action range A), the sheep/fox moves away from that shepherd. The sheep is otherwise passive. The active behaviour of the foxes is preprogrammed: each fox tries to intercept the sheep by estimating its future position and by heading in that direction. A trial ends when the sheep enters the corral or is captured by a fox. The experimental parameters of the task and the movement equations for the sheep and foxes are listed in Appendix A.4.

The fitness function rewards the shepherds for getting the sheep closer to the corral, and in case the sheep is successfully corralled, for the amount of time it took:

$$F_h = \begin{cases} 2 - \tau/T & \text{if sheep is corralled} \\ max(0, 1 - d_f/d_i) & \text{otherwise} \end{cases}, \tag{3.4}$$

where τ is the number of time steps elapsed, T is the maximum trial length, d_f is the final distance of the sheep to the corral, and d_i is the initial distance. The team-level behaviour characterisation $\beta_h(t)$ describes the effects of the shepherds on the sheep, while the agent-level characterisation $\beta_h(a)$ describes the role of shepherd, see Table 3.3.

TABLE 3.3: Behaviour characterisations used in herding task. All means are taken over the simulation time, and all features are normalised to the range [0,1].

Team-level characterisation $\beta_h(t)$	Individual-level characterisation $\beta_h(a)$
▷ Final distance of the sheep to the corral	▷ Mean distance of a to the sheep
▷ Mean distance of the sheep to the border of the arena	▷ Mean distance of a to the corral
▷ Mean distance between the sheep and the foxes	▷ Mean distance of a to *Fox 1*
▷ Trial length	▷ Mean distance of a to *Fox 2*

3.5.3 Evolutionary Setup

NEAT (Stanley and Miikkulainen, 2002) is used to evolve the neural network controllers (see Section 2.2). The parameters of the NEAT algorithm were the same for both tasks, and are listed in Appendix A.1. Novelty-driven cooperative coevolution was implemented over NEAT as described in Section 3.2, and with the same parameter values as the predator-prey experiments. In order to implement the NSGA-II algorithm in NEAT, the individuals were scored according to the Pareto front they belong and crowding distance, respecting the original NSGA-II ranking (Deb et al., 2002), and the selection and speciation processes relied on these scores (see Appendix A.1 for details).

3.5.4 Results

Figure 3.14 summarises the highest team fitness scores achieved in each evolutionary run, for each method and each task. Overall, the results obtained in the two tasks are consistent with the results obtained with the predator-prey task, presented in Section 3.4. In the herding task, *Fit* displays a very poor performance, and the best solutions consistently failed to drive the sheep towards the corral. In the cooperative foraging task, the performance of *Fit* displayed a very high variability: some runs achieved good solutions, where a reasonable number of items is collected, while others failed completely, with not a single item collected. In both tasks, *NS-Team* significantly outperforms *Fit* (Mann-Whitney test, adjusted $p < 0.001$). In the herding task, *NS-Team* consistently evolved solutions where the sheep was corralled (fitness above 1), and in the cooperative foraging task it consistently evolved solutions where at least three items were collected.

The relative performance of the novelty variants is also similar to the previous results. Novelty with team-level characterisations (*NS-Team*) displayed the highest performance in the cooperative foraging task ($p < 0.01$), and a similar performance to *NS-Mix* in the herding task ($p = 0.35$). Novelty based on individual-level characterisations (*NS-Ind*) was always significantly inferior to *NS-Team* (adjusted $p < 0.05$). The herding task was the only one where *NS-Mix* was able to match the performance of *NS-Team*. One possible reason for this result is that the herding task requires division of labour, rather than tight cooperation between the agents: each agent can

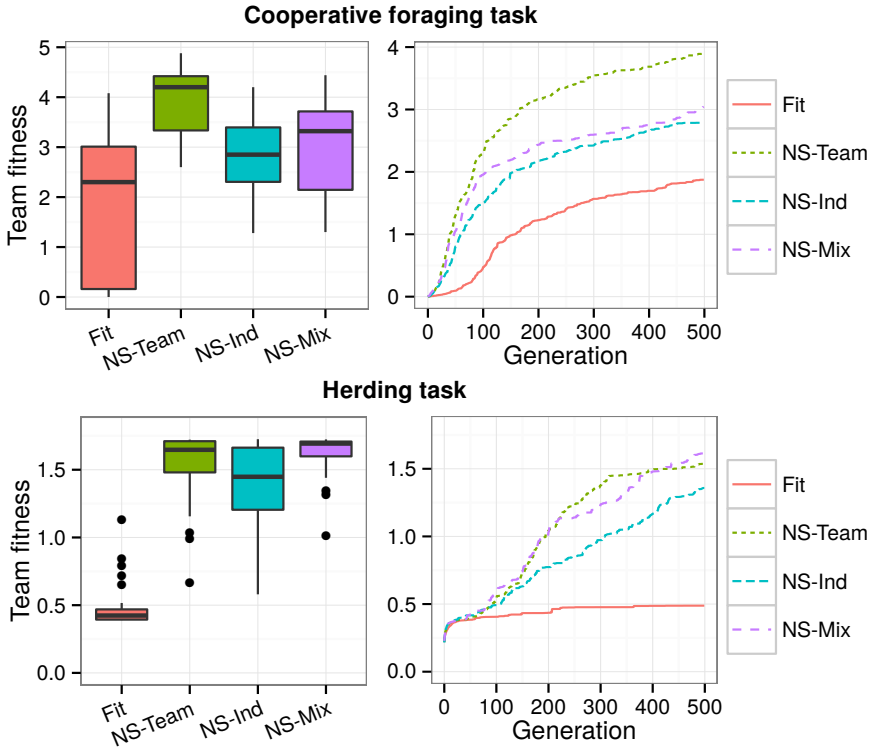


FIGURE 3.14: Left: highest team fitness scores achieved with each method and task. Each treatment was repeated in 30 independent evolutionary runs. The whiskers represent the highest and lowest value within 1.5 IQR. Right: highest fitness scores achieved so far at each generation, averaged over the 30 evolutionary runs.

perform its subtask independently, without relying on other agents, for example chasing one specific fox, or attempting to corral the sheep.

The analysis of the best-of-generation (BoG) teams (Figure 3.15) reveals that *Fit* fails in the herding task because it strongly converges to a very narrow region in the team behaviour space. In the cooperative foraging task, the problem of premature convergence is not so severe, as evidenced by the relatively high levels of BoG team dispersion, see Figure 3.16. The performance of *Fit* was, however, significantly inferior to the other methods that obtained similar values of BoG team dispersion. The results in Figure 3.15 (top) suggest an explanation for this phenomenon: although *Fit* achieves a fair amount of behavioural exploration, the exploration is focused on a region that is distant from high-quality solutions (bottom-right corner of the space).

In both tasks, *NS-Team* and *NS-Mix* display the highest levels of team behaviour dispersion, considering all teams ($p < 0.001$). *NS-Ind* has relatively high levels of individual behaviour dispersion in both tasks, but they neither translate into a higher diversity of team behaviours, nor the achievement of higher quality solutions.

3.6 Discussion

Premature convergence to stable states Our results obtained with a simple genetic algorithm in the predator-prey task first showed that fitness-based coevolution (*Fit*) often fails as the task becomes more complex. The populations often converge to suboptimal equilibria, and therefore fail to achieve effective solutions for the task. In Section 3.5, we tried fitness-based coevolution with a more elaborate neuroevolution algorithm — NEAT, that sustains high genetic diversity in the populations. We

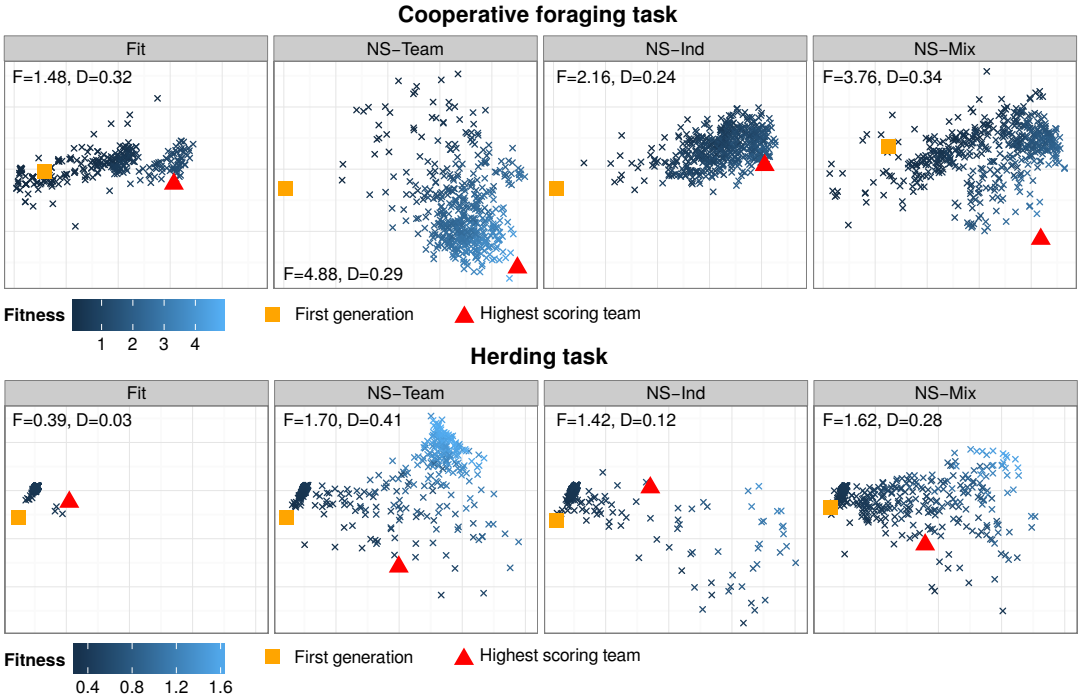


FIGURE 3.15: Behaviour of the best-of-generation teams in representative evolutionary runs. The behaviour space was reduced to a two-dimensional space with Sammon mapping.

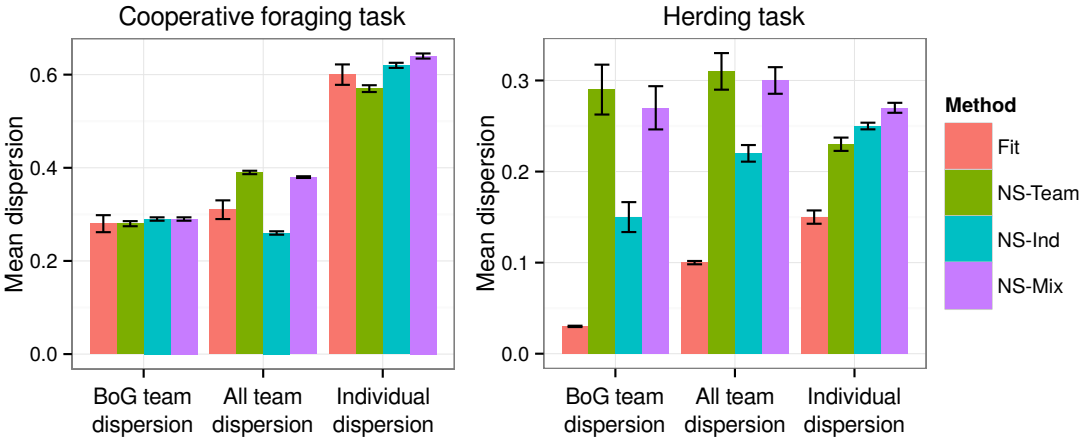


FIGURE 3.16: Mean dispersion of the best-of-generation teams, team behaviour exploration, and individual behaviour exploration (see Section 3.3.1) for each evolutionary setup. The respective standard error bars are shown.

experimented with two additional tasks: cooperative foraging and herding. Even with higher genetic diversity in the populations, fitness-based coevolution often converged prematurely in these tasks. As previous works have shown (Panait et al., 2006b; Wiegand, 2003), premature convergence is not necessarily caused by lack of genetic diversity, but by a strong attraction to stable states: populations can become over-adapted to one another. The issue is not related to the evolutionary algorithm itself, but to the way the population individuals are rewarded in a coevolutionary algorithm.

As suggested in previous works (Panait, 2010), we increased the number of collaborations with which an individual is evaluated, to increase the likelihood of convergence to (near-)optimal solutions. This strategy, however, only worked in the

two-population setup. In the three-population setups, increasing the number of random collaborations failed to improve the performance of fitness-based coevolution. Our results showed that increasing the number of collaborations does not help the coevolutionary algorithm to escape stable states.

Novelty-driven cooperative coevolution To overcome convergence to suboptimal equilibria, we proposed to add a novelty score in the evaluation of the individuals of each population. We assessed three cooperative coevolutionary algorithms based on novelty search, each with a different way of computing the novelty scores: (i) novelty based on team-level behaviour characterisations (*NS-Team*), (ii) based on agent-level characterisations (*NS-Ind*), and (iii) a combination of the two (*NS-Mix*). In all methods, we used a multiobjective algorithm, NSGA-II, to combine the novelty and team fitness objectives. In the case of *NS-Mix*, three objectives were used: individual novelty, team novelty, and team fitness.

Our results clearly revealed that the most effective way of introducing novelty search in CCEAs is *NS-Team*. The relative performance of the novelty-based methods was consistent across all the considered task setups: *NS-Team* $>$ *NS-Mix* $>$ *NS-Ind*. The algorithms based on individual-level evaluations (*NS-Ind* and *NS-Mix*) could evolve more diverse agent behaviours, but typically this did not translate to more diverse or effective team solutions. Our results suggest that encouraging novelty of agent behaviours can actually be harmful for the adaptation of the populations to one another. *NS-Ind* was always the lowest performing novelty-based method.

When compared to fitness-driven coevolution, *NS-Team* evolved significantly better solutions for almost all task setups. The more challenging the task setup was, the greater the performance difference between *NS-Team* and *Fit*, as *NS-Team* successfully managed to avoid convergence to stable states. *NS-Team* could also discover a greater diversity of team behaviours in a single evolutionary run. In the predator-prey task, we showed that *NS-Team* evolved a diverse set of solutions for the task, whereas *Fit* tended to focus on a single class of solutions.

Scalability with the number of agents In the predator-prey task, we evaluated *NS-Team* in task setups varying from two to seven agents, with each agent evolving in a separate population. *NS-Team* scaled well with the number of agents, evolving good solutions for all team sizes. For the same task setup, increasing the number of predators never harmed the performance of *NS-Team*. Our analysis revealed that *NS-Team* can take advantage of most of the available agents to solve the task, even when a lower number of agents is actually enough, which suggests that *NS-Team* can evolve cooperation for a relatively large number of agents.

In future work, we will evaluate the proposed approach with larger multiagent systems. One concern is that with relatively large teams, one particular agent might not have a significant impact in the behaviour of the team as a whole, thus resulting in less accurate fitness and novelty gradients. In Chapter 6, we address this issue by proposing the evolution of partially heterogeneous teams.

Parameter sensitivity and generalisation When using novelty-based techniques, the experimenter must provide a behaviour similarity measure. For each of the considered tasks, we chose a small number of behavioural traits that intuitively described the behaviour of the agents in the context of the task objective. The chosen behavioural traits were based on *systematically derived behaviour characterisations* (Gomes et al., 2014e): they were always directly observable in the task, and did

not require complex calculations and/or fine tuning. Although the definition of behavioural measures did not pose a problem in these tasks, we have shown, along with other authors, that it is possible to avoid the use of manually specified behaviour measures by relying on generic measures (Gomes and Christensen, 2013; Meyerson et al., 2016; Mouret and Doncieux, 2012, see Section 2.5.4).

Our tasks were based on two different neuroevolution algorithms: a simple genetic algorithm with direct encoding and no crossover, and NEAT (Stanley and Miikkulainen, 2002), a neuroevolution algorithm with topology evolution, crossover, and fitness sharing. Novelty-driven coevolution performed well with both algorithms, and the relative performance of the methods was consistent, which suggests that the proposed methods are independent of the underlying evolutionary algorithm.

Besides the experiments reported in this chapter, we have also evaluated an early version of novelty-driven cooperative coevolution in a keepaway soccer task (Gomes et al., 2014a). In this task, we evolved a team of three *keepers* that have to cooperate to keep the possession of the ball, against a pre-programmed *taker* that goes after it. The evolved controllers were fixed-topology neural networks. We evaluated fitness driven coevolution (*Fit*), and an early version of *NS-Team* and *NS-Ind*[3]. The results were highly consistent with the experiments reported in this chapter: (i) *NS-Team* achieved significantly higher fitness scores than *Fit*; (ii) *NS-Ind* yielded very poor results; and (iii) *NS-Team* explored the team behaviour space much more than *Fit* and *NS-Ind*.

3.7 Summary

In this chapter, we showed that rewarding individuals that cause novel team behaviours (*NS-Team*) is a promising approach to avoid convergence to suboptimal equilibria. *NS-Team* consistently outperformed traditional fitness-driven coevolution across multiple task setups, achieving higher team fitness scores and a wider diversity of effective solutions. The proposed approach only requires one collaboration to evaluate each individual, which contrasts with previous approaches that relied on using a large number of collaborations to overcome premature convergence. *NS-Team* is therefore a convenient approach for overcoming premature convergence in problem domains where the evaluations are costly, namely multirobot systems. We also show that *NS-Team* is compatible with more than two populations, contrasting with previous approaches that are studied only for two-population coevolutionary algorithms. In the following chapters, we will further assess *NS-Team* by studying how it performs in significantly different domains: (i) a multirobot task performed on real robots; and (ii) a multirobot task where radically different robot types need to cooperate.

[3]Compared to the novelty-driven coevolution proposed in this chapter, the early versions proposed in (Gomes et al., 2014a) had the following differences: (i) novelty was combined with fitness with a linear scalarisation of the scores, with an equal weight of novelty and fitness; and (ii) two collaborations were used to evaluate every individual, one formed by the highest scoring individuals of the other populations, and other formed by randomly picked individuals of the other populations.

Chapter 4

Validation in a Real Multirobot System

As discussed in Section 2.4.4, previous works that have applied CCEAs to the evolution of agent behaviours can be divided in three main categories (Panait and Luke, 2005a): (i) *non-embodied agents*, including static function optimisation (Potter and De Jong, 1994) and evolutionary game theory (Wiegand et al., 2002); (ii) *abstract embodied agents*, where the evolved agents are situated in an environment that they sense and act in, but the agents are high-level abstractions and unrelated to any real robotic platform (Potter et al., 2001; Yong and Miikkulainen, 2009), which includes the experiments presented in the previous chapter; and (iii) *simulated robotics tasks*, in which the evolved agents are modelled closely after a real robotic platform and a real task environment (Nitschke, 2005a; Nitschke et al., 2012b). One notable category missing from this list is *real robotics tasks* – tasks in which behavioural control is evolved in simulation, and then transferred to a real robot team. While this *reality gap* has been crossed using other evolutionary algorithms (Silva et al., 2016b), in both single (Jakobi, 1997) and multirobot systems (Duarte et al., 2016b), to the best of our knowledge, robotic control evolved with CCEAs has been confined to simulation-based experiments up until now.

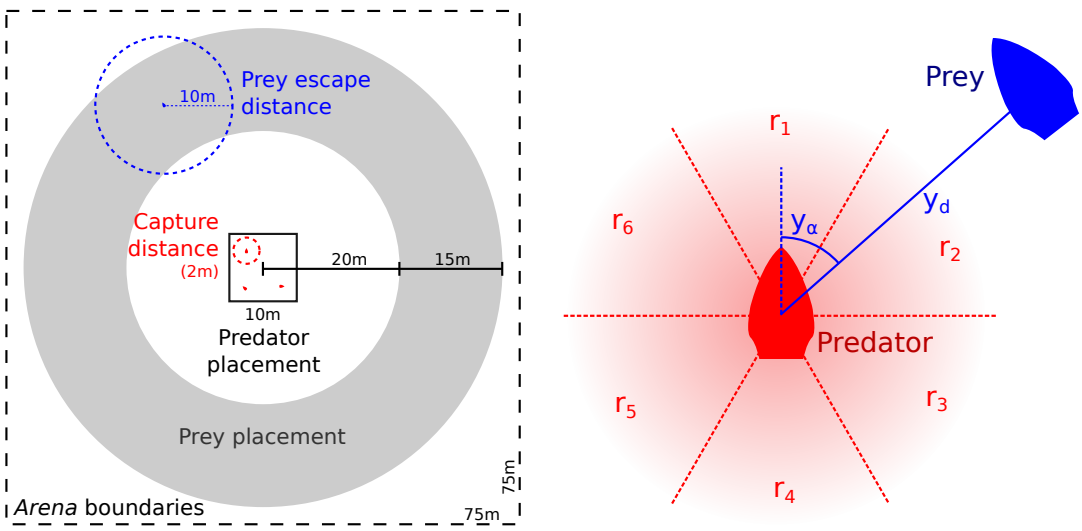
The reality gap (Jakobi, 1997) is a central issue with the simulate-and-transfer approach. Controllers evolved in simulation can become ineffective once transferred to the physical robots because of their exploitation of features of the simulated world that are different or do not exist at all in the real world. Overall, the difficulty of accurately simulating physical systems is well known in robotics (Matarić and Cliff, 1996). Differences between simulation and the real world include inaccurate sensor modelling, and simulation-only artifacts caused by simplifications, abstractions, and discreteness of physics implementations. In evolutionary robotics, the reality gap is a frequent phenomenon and one of the main impediments for progress (Koos et al., 2013). Validating evolutionary robotics techniques in real hardware is thus pertinent, as it represents the ultimate test to their effectiveness (Silva et al., 2016b).

In this chapter, we extend the experiments presented in Section 3.4 to a real robotic platform, with the objective of validating and comparing in a real system both the standard fitness-driven CCEA as well as the proposed novelty-driven cooperative coevolution approach. We present experiments where we evolved control for an aquatic surface multirobot system that must perform a predator-prey task. After evolving the controllers offline in simulation, the controllers are transferred to the real robotic platform, and systematically evaluated in an outdoor environment. The natural unpredictability associated with the aquatic environment (caused by inaccurate robot motion, waves, currents, and so on) allow us to study transferability in a realistic scenario, and understand how controllers evolved by cooperative coevolutionary algorithms are able to cope with noisy and stochastic conditions.

4.1 Aquatic Predator-prey Task

In our aquatic predator-prey task, a team of three predators must cooperate to capture one escaping prey. Similarly to the task used in Section 3.4, only the controllers of the team of predators are evolved, while the prey has a pre-specified fixed behaviour. We had to adapt the experimental setup in order to be compatible with the limitations imposed by the aquatic environment and the real robots used. In particular, the initial conditions are significantly more stochastic, as in an aquatic environment it is not practical to initially place the predators at fixed locations (as done in Section 3.4), and the predator robots have additional sensors that allow them to sense the other predators, as relying on a fixed strategy to catch the prey (as previously done) would not be possible in such stochastic conditions.

In each trial, the three predators are placed in the centre of the arena, with random positions and orientations (Figure 4.1a), while the prey is placed in a random location away from the centre. A trial ends if a predator gets close to the prey (less than 2 m), if the prey escapes the arena, or if the time limit of 75 s is reached. The prey tries to escape in the opposite direction of the closest predator, if that predator is closer than 10 m. The prey can move up to the maximum possible speed of the predators, meaning that the predators typically cannot outrun it. Cooperation among the predators is therefore essential to capture the prey. The parameters of the experimental setup are listed in Appendix A.5.



(a) Setup of the simulated task.

(b) Predator sensory inputs.

FIGURE 4.1: Illustration of the task setup used for the evolutionary process, and the predators' sensory inputs (used both in simulation and in the real robots).

4.2 Robotic Platform

The experiments are based on an aquatic multirobot system (Costa et al., 2016) that had been previously used in other evolutionary robotics studies (Duarte et al., 2016a,b,d,e). Each robot (Figure 4.2) is a small (65 cm in length) differential drive mono-hull robot, equipped with GPS and compass. The robots continuously broadcast their position to the neighbouring robots using Wi-Fi. This data is then parsed locally by the robots to calculate the sensory inputs. The same robotic platform is

FIGURE 4.2: Photo of one robot in the water. The robots are autonomous surface vehicles equipped with Wi-Fi for communication, and a compass and GPS for localisation.

used for both the predator robots and prey robot. For practical reasons, the predators sense the prey the same way they sense each other, i.e., the prey robot also broadcasts its position to the nearby predators. This allowed us to avoid hardware challenges related to the detection of foreign objects on the water surface, which are out of the scope of this study. Additional details of the robotic platform are presented in Appendix A.5 and in (Costa et al., 2016).

Each predator robot relies on the following sensory inputs, which are calculated from the location data received by the broadcasts of the other robots, and its own position and orientation (Figure 4.1b):

Predator sensing $(r_{1..6})$**:** Six inputs for detecting the other predators, corresponding to six equally-sized circular sectors around the robot. Each input corresponds to the distance to the closest predator in the corresponding sector, or the maximum value if no predator is present there.

Prey location $(y_{\alpha,d})$**:** Two inputs returning (i) the relative angle from the predator to the prey (zero corresponds to straight ahead), and (ii) the normalised distance from the predator to the prey. If the prey is not within sensing range, the sensors return an angle of zero and the maximum distance.

The sensory input values (all scaled to $[-1, 1]$) are fed to the neural network controller, which outputs the desired linear speed and angular velocity of the robot. These two output values are then converted to left and right motor speeds and applied to the robot's motors.

4.3 Evolutionary Setup

4.3.1 Simulation Approach

In order to ensure a successful transferability of the controllers from simulation to reality, it is paramount to adequately configure the simulation environment in which the controllers are evolved. A number of different approaches have been proposed in the past to improve transferability (Silva et al., 2016b). One of the most widely used approaches was proposed by Miglino et al., (1995), which consists of three complementary techniques: (i) using samples from the real robots' sensors to more accurately model them in simulation, (ii) introducing a conservative form of noise to promote the evolution of robust controllers, and (iii) continuing evolution in real hardware to tune controllers to the differences between simulation and reality. The

use of noise in simulation promotes the evolution of general and adaptive solutions, therefore increasing the performance of transferred solutions (Jakobi, 1997; Miglino et al., 1995). The introduction of noise is a computationally-effective way of promoting the evolution of robust controllers, since it becomes more difficult to exploit particular features of the simulator.

In this study, we followed the *conservative noise* approach as done in (Duarte et al., 2016b). We used a two-dimensional simulation environment, where the robots were abstracted as circular objects with a certain heading and position. The general principle behind the simulation was to model the motion of the robots based on real measurements taken in the water, but without including physics simulation and fluid dynamics, which would have resulted in a complex and computationally expensive simulation environment. We injected considerable amounts of noise in the sensors and actuators based on measurements taken from real robots, and the initial task conditions were significantly varied in every simulation trial. Every team was evaluated in 10 simulation trials.

The variation of the initial conditions (detailed below) accounts for the stochasticity of the task, and the minor differences in hardware from robot to robot (caused for example by different battery levels, motor conditions, and sensor calibrations). The noise injected at every control step, on the other hand, accounts for the imprecision of the real sensors and the uncertainty of movement in an aquatic environment. The following parameters were varied during the simulation trials (see details in Appendix A.5):

Initial conditions: the maximum speed of each motor was independently varied; the compass sensor was slightly offset by a random amount; the prey's escape speed was set to 75% to 100% of the predators' regular maximum speed; and the initial positions and orientations of all robots were varied according to Section 4.1.

Every control step: GPS and compass with reading errors according to the hardware specification; the real motor output can be different from the desired speed; and the prey's movement direction varied slightly from the calculated direction (away from the nearest predator).

It should be noted that the noise was not injected directly into the sensory inputs and actuator outputs of the neural controller. Instead, it was injected on the simulated hardware sensors and actuators, which in turn are closely dependent on the inputs and outputs of the controller. This allowed us to model the noise in a way that is more similar to the noise that affects the real robots.

4.3.2 Evolutionary methods

Both fitness-driven and novelty-driven cooperative coevolution followed the implementation used in the experiments in Section 3.5: the populations used NEAT (Stanley and Miikkulainen, 2002) as the underlying neuroevolution algorithm, and in novelty-driven coevolution the novelty scores are combined with the fitness scores with a multiobjective Pareto ranking. The full list of parameters can be found in Appendix A.1. The fitness function is the same as the one used in the experiments in the previous chapter, see Equation 3.3, as well as the team behaviour characterisation, see Table 3.1.

4.4 Evolving and Identifying Diverse Solutions

4.4.1 Quality of Solutions

Each evolutionary approach was repeated in ten independent evolutionary runs. To obtain a more accurate estimate of the evolved teams' quality and behaviour, all the *best-of-generation* teams (Definition 7) were re-evaluated a posteriori in 50 simulation trials. The fitness scores achieved are shown in Figure 4.3. On average, the evolutionary runs of *Fit* achieved a highest fitness score of 1.09 ± 0.10, and *NS-Team* achieved 0.96 ± 0.20. While this difference was marginally significant ($p = 0.043$, Mann-Whitney U test), both approaches managed to evolve high-quality solutions with fitness scores above 1.0, meaning the prey was captured.

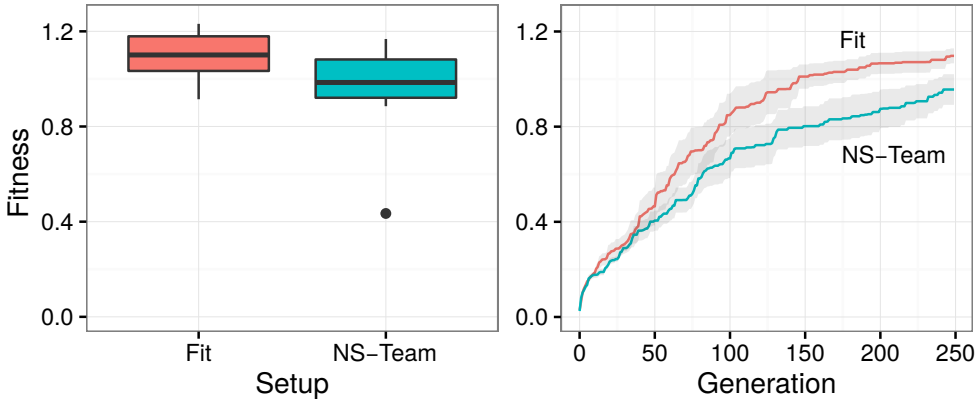


FIGURE 4.3: Left: highest fitness scores achieved with each method in each evolutionary run. Right: highest fitness scores achieved so far at each generation, averaged over the ten evolutionary runs for each method.

The high fitness scores and consistent results obtained with *Fit* suggest that premature convergence is not a significant issue in this task. This explains why *Fit* slightly outperforms *NS-Team* with respect to the highest fitness scores achieved — *Fit* can simply follow the fitness gradient, while *NS-Team* is simultaneously trying to maximise the behavioural novelty objective. The potential advantages of using *NS-Team* in this task are therefore restricted to the diversity of solutions.

4.4.2 Behavioural Diversity

We first analyse the behaviour exploration according to the measures proposed in Section 3.3.1, see Figure 4.4. The results show that *NS-Team* achieved a significantly higher behaviour exploration than *Fit*, considering all the teams evolved ($p = 0.005$, Mann-Whitney), but the exploration considering only the *best-of-generation* teams was not significantly different ($p = 0.12$). This last result supports the previous conclusion that premature convergence is not a significant issue in this task – *Fit* is able to successfully evolve good teams and progressively refine them, thus exploring the behaviour space.

To visualise the diversity of behaviours evolved by each evolutionary approach, we mapped the *best-of-generation* teams according to their behaviour characterisation vector[1], using a Kohonen map for reducing the dimensionality (see Section 3.3.1).

[1]The Kohonen map visualisation only used the *best-of-generation* teams since they were the only ones that were re-evaluated in additional simulation trials after evolution. Given the highly stochastic nature of this task, relying on the behaviours recorded during evolution for the selection and transfer of a small number of solutions could yield inaccurate results.

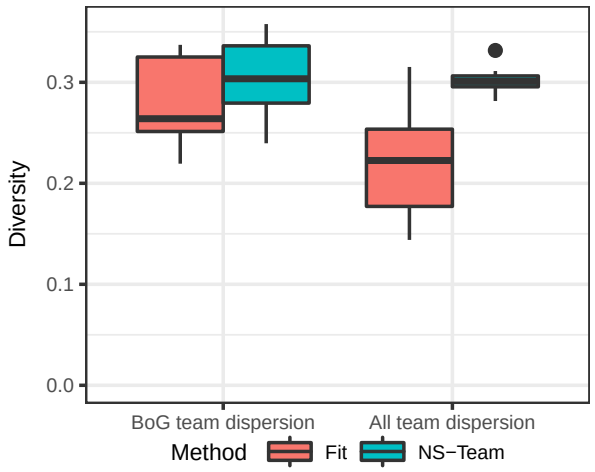


FIGURE 4.4: Analysis of the exploration of the behaviour space in the evolutionary runs, using the dispersion of the *best-of-generation* teams (Definition 8) and the dispersion of all the evolved teams (Definition 9).

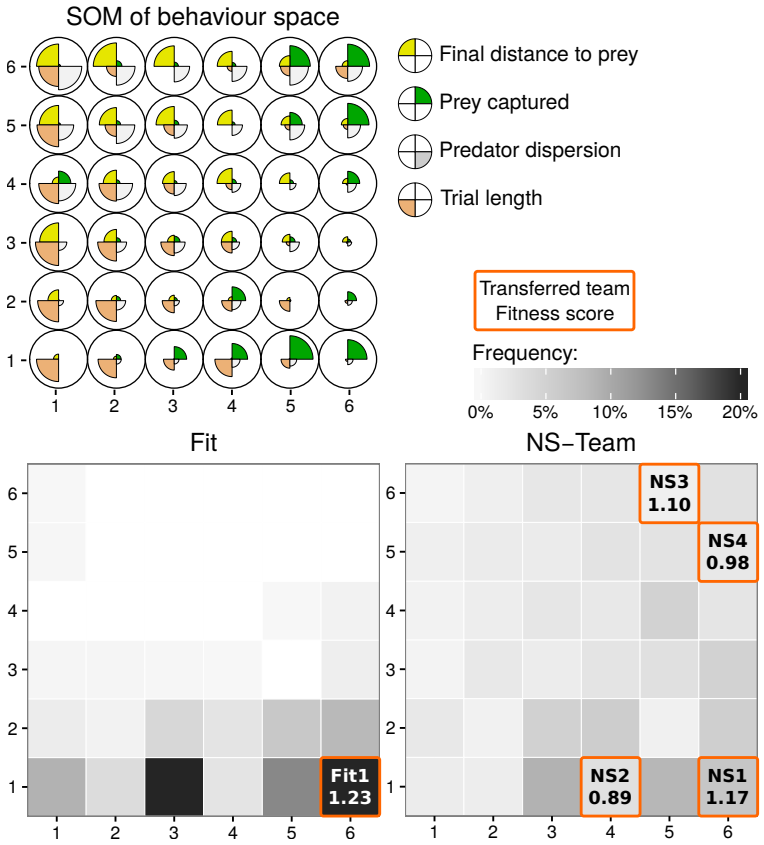


FIGURE 4.5: Left: trained Kohonen map, where each node represents a region of the team behaviour space. Middle and right: team behaviour exploration by the two evolutionary approaches. The darker a region, the more of the evolved teams belonged to it.

In Figure 4.5 we show the diversity of teams evolved in all the evolutionary runs of each method. The results show that *NS-Team* could reach behaviour regions that were never reached by *Fit*, which is consistent with the results reported in Chapter 3. Based on these results, we then proceeded to select a diverse set of solutions to be tested in the real robots. We selected different regions of the behaviour space where

the prey capture rate was high, and identified the team belonging to each of those regions that obtained the highest fitness score. We chose one team evolved by *Fit*, as all the high-quality teams were found in the bottom-right corner of the map, and four solutions evolved by *NS-Team*, from different regions of the map with high *prey capture* values.

4.5 Transferring the Teams to Real Robots

The selected teams were then evaluated in the real multirobot system, using the neural network controllers exactly as they had been evolved. The experiments were performed in a semi-enclosed water body, see Figure 4.6. The task setup was similar to the setup used during evolution (Section 4.1): the three predators were placed close to the centre of the arena, and the prey was placed at approximately 25, 30, and 35 m away from the centre. Each team was assessed in three independent trials, with each trial lasting for at most 100 s. The arena boundaries were 100×100 m, and the prey moved at the maximum speed (the same as the predators' maximum speed). To compare the results of the real-robot experiments with simulation, the chosen teams were re-evaluated in 500 simulation trials, using the same initial conditions as the real-robot experiments. The fitness scores and behaviour features of the teams operating in the real environment were computed using the GPS data logged by the robots.

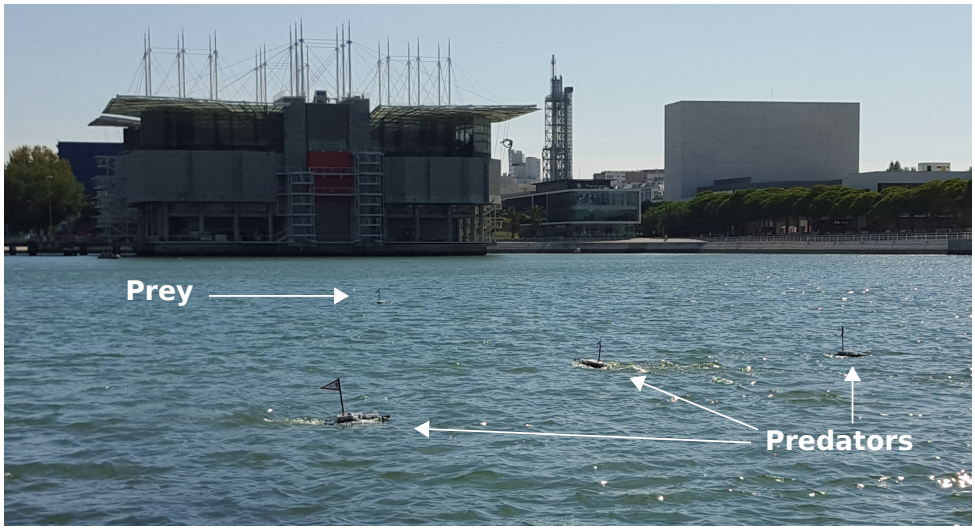


FIGURE 4.6: Photo of the real-robot experiments, at Parque das Nações, Lisbon, Portugal, in a semi-enclosed area in the margin of the Tagus river.

In Figure 4.7 (Fitness), we compare the fitness scores obtained by the teams in simulation and in the real robots. We additionally explore the diversity of team behaviours by comparing the controllers' performance in reality and in simulation according to the behaviour features that were used in novelty-driven coevolution. The results show that all teams except *NS2* were able to capture the prey in the majority of the trials. The fitness scores obtained in the real experiments are similar to the scores obtained in simulation, fitting in the distribution obtained in simulation. These results are a first indication that the evolved controllers were generally able to cross the reality gap successfully.

The effectiveness of the team behaviours was confirmed by analysing the traces of the robots' movement in the real-robot experiments, shown in Figure 4.8. The

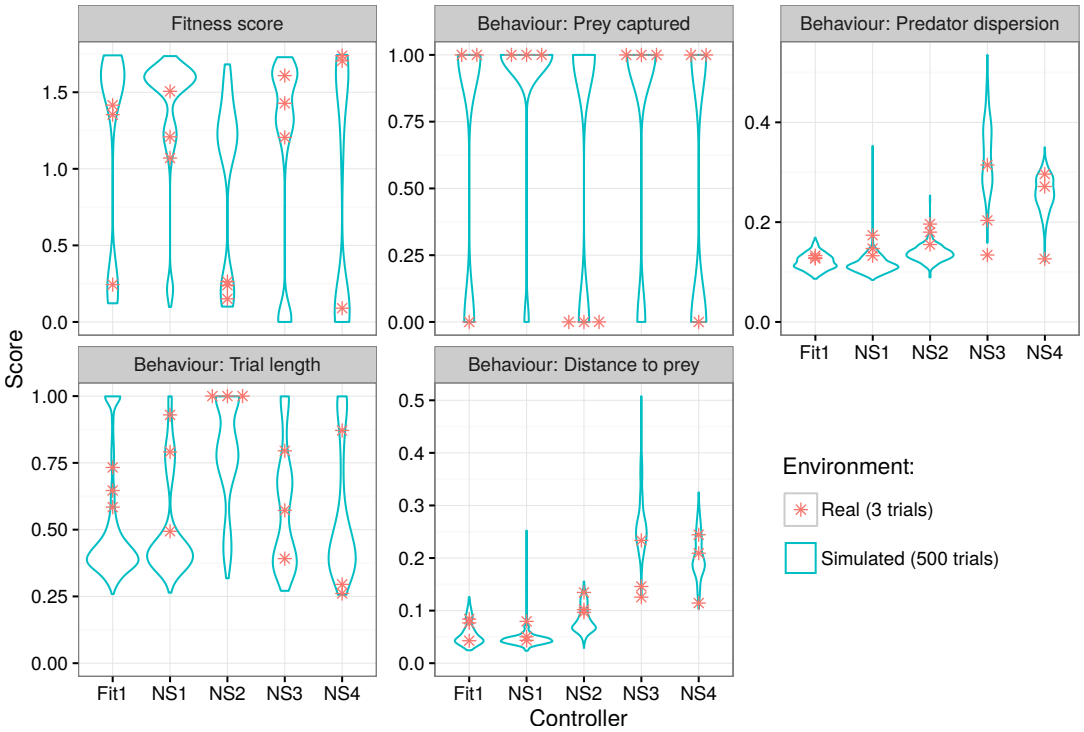


FIGURE 4.7: Comparison of the fitness score and behaviour features obtained in the real-robot experiments (asterisks) and in simulation (violin plots) in similar conditions.

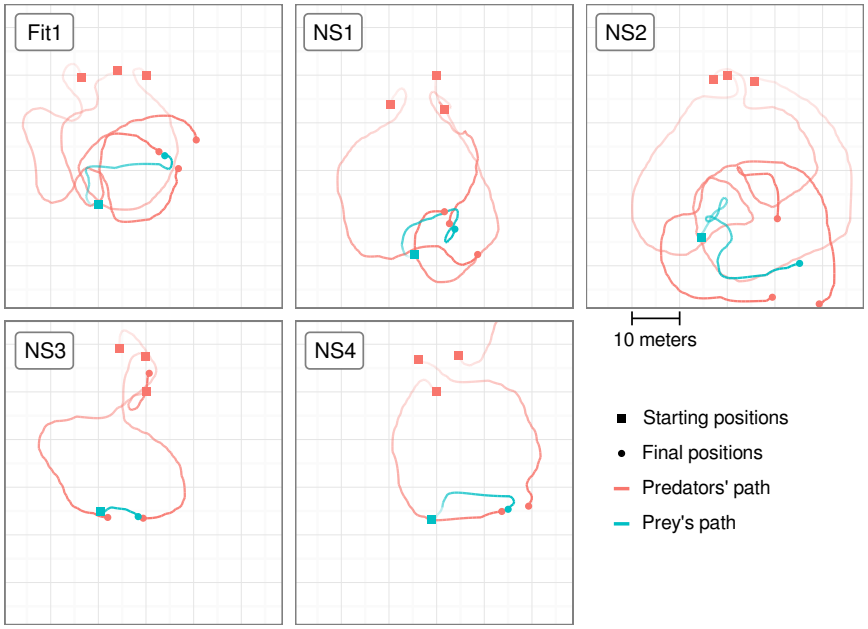


FIGURE 4.8: Traces of one experimental trial (out of three) for each of the teams evaluated in the real robots. Traces and videos of all real-robot experiments are available online: `https://doi.org/10.5281/zenodo.49582`.

Fit1 and *NS1* teams displayed a behaviour where the three predators would initially spread and move towards the prey, each approaching the prey from a different direction. The behaviour of *NS2* was similar to *Fit1* and *NS1*, but the predator team dispersed more. The teams *NS3* and *NS4* displayed a significantly different behaviour:

only two predators chased the prey, approaching it from opposite directions, while the remaining predator would move away from the group. The observed robot traces are consistent with the measured behavioural features (Figure 4.7), and confirm that novelty-driven coevolution was able to achieve a wide diversity of team behaviours. For instance, it is possible to observe that *NS3* and *NS4* display a higher dispersion and final distance to prey, which is explained by the fact that in these teams, only two predators chase the prey. The differences and similarities between the team behaviours observed in the real-robot experiments are consistent with the behaviour map obtained in simulation (Figure 4.5).

Overall, despite the stochastic conditions of the aquatic environment, the predators displayed effective cooperation, and were consistently able to solve the task. The team of predators would often fail to capture the prey in the first attempt, but the team would then spread out and try to encircle the prey again. Moreover, robots sometimes displayed temporary motor failures (see online videos), which did not compromise the effectiveness of the team. These behaviours suggest that the teams were not overfitted to the simulation environment, and could effectively adapt to different scenarios.

4.6 Discussion

In this chapter, we validated novelty-driven cooperative coevolution in a task where the controllers were evolved in simulation and then transferred to real robots, and compared it with the traditional fitness-driven CCEA. The task was based on the predator-prey task used in Chapter 3, with the necessary modifications for this specific environment and robots. The evolutionary processes were conducted exclusively in simulation, and a number of high-fitness teams were then systematically evaluated in real robots operating in a non-controlled outdoor aquatic environment.

The evolved teams generally transferred well to the real robots, successfully crossing the reality gap. Out of the five teams tested, four teams could consistently capture the prey, and obtained fitness scores very similar to those obtained in simulation. The cooperation between robots that was exhibited in simulation was also observed in real robots, and the teams displayed robust behaviours that did not appear to be overfitted to the simulation environment. It should be noted, however, that the task setup (including initial distance of predators to prey, arena size, and so on) of the real-robot experiments was very similar to the task setup that was used in the evolutionary process. Additional experiments would be required to assess if the evolved behaviours can generalise to different task variants.

The successful transfer is especially notable given that we used low-fidelity simulator during evolution, and given the stochastic nature of the real task environment. We encouraged the evolution of robust and transferable controllers by introducing conservative amounts of noise and variations in the sensors and actuators of the robots in simulation, and by using multiple trials to evaluate each solution, with different initial conditions. The noise injected in the task also simulated small morphological differences among the robots (such as different speeds), therefore encouraging the evolution of controllers that are robust to variations in the behaviour of the team members. This strategy revealed to be successful in producing transferable controllers; it was relatively simple to implement; and did not require additional fine-tuning, as the noise parameters are based on the actuator/sensor errors that were observed in the real robots or described in the hardware components' specification.

Novelty-driven cooperative coevolution was able to produce a good diversity of high-quality team behaviours for solving the task, which were identified following a systematic approach: the behaviour space was divided in discrete regions using a self-organised Kohonen map, and we then selected the highest-fit teams belonging to different behaviour regions. The diversity of behaviours that was observed in simulation was also present in the real multirobot system, which was confirmed by both the observation of the robot trajectories, as well as the extracted behaviour features.

4.7 Summary

We demonstrated that CCEAs can be successfully used to synthesise control for a real multirobot system, operating in an environment outside controlled laboratory conditions. Our experiments also demonstrated the potential of novelty-driven cooperative coevolution in real robots, and confirmed it as a valuable approach to evolve diverse team behaviours. Despite the large number of previous works that have showed the potential of CCEAs for evolving heterogeneous multirobot systems, this work stands amongst the first to demonstrate this potential in real robots and in a realistic environment. In the next chapter, we will validate CCEAs in another prominent type of multirobot systems: morphologically heterogeneous systems.

Chapter 5

Morphologically Heterogeneous Systems

One theoretical benefit of the CCEA architecture is that since populations are isolated, it is possible for different populations to evolve radically different agents, with genomes of different size, using different encodings, and even different genetic operators. This possibility of arbitrary heterogeneity has, however, only been exploited to a limited extent. In the domain of multirobot systems, most previous works only use CCEAs to evolve controllers for behaviourally heterogeneous, but morphologically homogeneous, multiagent systems, see Section 2.4.4. This means that all agents in the system have a similar complexity, similar morphological capabilities, and use the same genotype representation — the heterogeneity is exclusively at the behavioural specialisation level.

There have only been a few reports of successful evolution of morphologically heterogeneous systems, and in these studies, agents had only minor morphological differences, for instance: we used a *keepaway soccer* task where agents have slightly different moving and passing speeds (Gomes et al., 2014a); Yang et al., (2012) solved a *foraging* task where similar agents have different movement speed and sensing ranges; Blumenthal and Parker, (2004) solved a *predator-prey* task where the predators have slightly different linear and turning speeds; and Knudson and Tumer, (2010) use a *multi-rover target observation* task where there the heterogeneity is artificial – there are two different robot types, and the robots can distinguish between them, but all robots have the same capabilities.

In this chapter, we explore a higher degree of heterogeneity, evaluating for the first time the potential of CCEAs to evolve control for multirobot systems where there is a radical morphological heterogeneity within the team. In such morphologically heterogeneous systems, robots have significantly different actuation and sensing capabilities, and collaborate to take advantage of the collective set of capabilities (Dorigo et al., 2013). Morphologically heterogeneous systems have shown their value in many real-world domains of application, as discussed in Section 2.1.2.

A key element in the evolution of cooperative behaviours is *synchronised learning* (Uchibe et al., 1998): populations should exhibit a mutual development of skills, in order to avoid loss of fitness gradients and convergence to mediocre stable states, as discussed in Section 2.4.2. One concern when applying CCEAs to such radically heterogeneous systems is that since populations have to evolve largely different controllers, with different complexity, synchronised learning might be less likely to occur, potentially causing convergence to mediocre stable states and/or loss of fitness diversity. In this chapter, we study this effect and how this issue can be mitigated. Our experiments are based on a simulated foraging task where a ground

robot with very limited capabilities must cooperate with an aerial robot with a significantly more complex sensor-effector configuration. We explore several task variants to study how the differences between the robots, regarding the skills that must be evolved, and the difficulty in achieving cooperation, affect the performance of cooperative coevolution. We assess how novelty-driven cooperative coevolution (Chapter 3) can help mitigate this problem, and compare it with other competing techniques.

5.1 Aerial-ground Foraging Task

We use a task based on the cooperative foraging task presented in Section 3.5.1. Unlike the cooperative foraging task, however, the two robots have different capabilities – one ground robot can capture the items, while the other robot (aerial) has the capability to efficiently detect the items, see Figure 5.1. The ground robot has significantly fewer sensory capabilities than the aerial robot (detailed below). The two robots cannot communicate explicitly: they can only sense the relative position of each other when in close proximity. To accomplish the task, the robots have to establish a tight cooperation – the aerial robot must learn to find the ground robot, and then guide it towards collectable items in the environment. Complementary, the ground robot should follow the aerial robot and collect the items found.

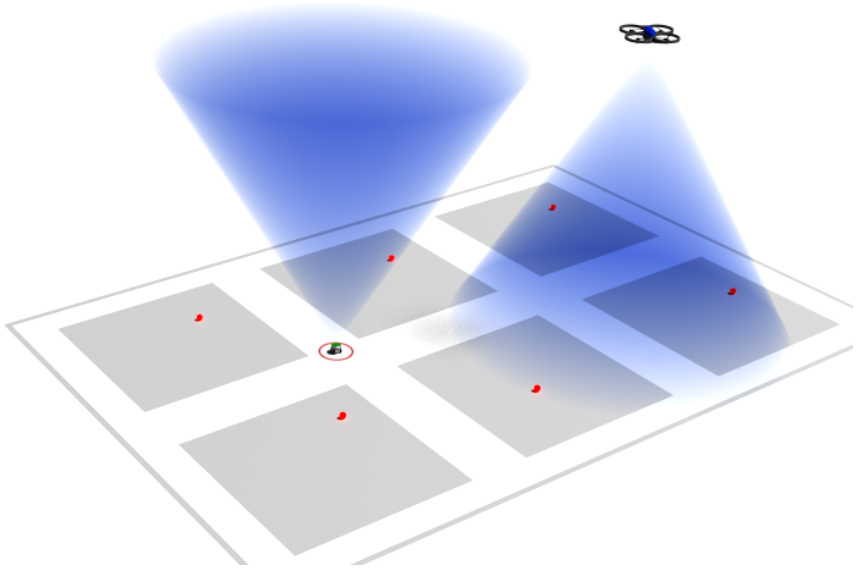


FIGURE 5.1: Illustration of the aerial-ground foraging task, during task execution. The red spheres are the items to be collected. One item is placed in each of the grey zones. The blue cones depict the viewing range of the robots. The red circle around the ground robot depicts the range of its item sensor.

5.1.1 Robot Configurations

Both robots have sensors modelled after a vertical camera sensor, facing up and down for the ground robot and aerial robot, respectively. The camera has a field of view of 60°, and can detect objects up to a vertical distance of 250 cm, see Figure 5.1. The camera image is not directly used by the robots' evolved controller: the sensor cone is divided in equally-sized sectors, and the value of each input is the presence

or distance of the object in the respective circle sector (Figure 5.2). The robots rely on these camera-based inputs for detecting each other. In the case of the aerial robot, camera-based inputs are also used to detect the items that must be collected. Only the ground robot has the ability to collect items from the environment. To collect an item, the robot simply has to pass over it. The ground robot is a small differential drive robot, modelled after an *e-puck* (Mondada et al., 2009), while the aerial robot is modelled after a quadcopter. The sensory inputs and actuators available to each robot are listed in Table 5.1.

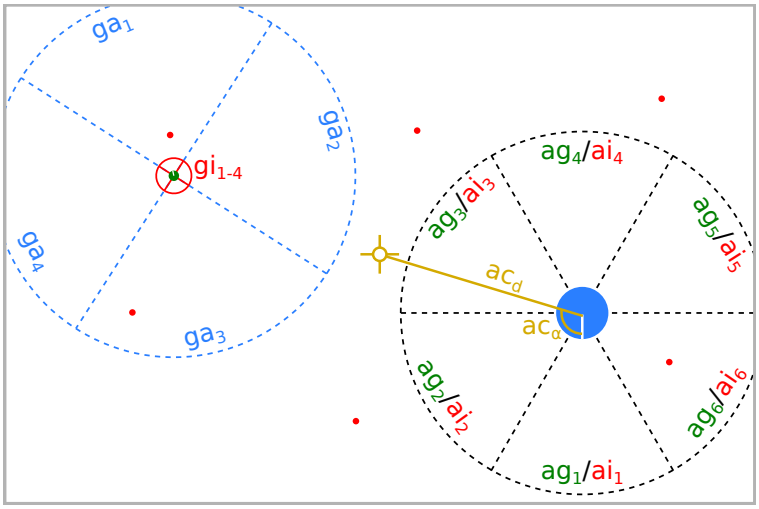


FIGURE 5.2: Illustration of the robots' sensors. See Table 5.1 for the sensors description. Note that the inputs **ga**, **ag**, and **ai** are modelled after a camera sensor, and therefore its horizontal range varies depending on the aerial robot's current altitude.

5.1.2 Task Variants

To study how cooperative coevolution is affected by differences in the learning speed of the agents, and the difficulty of establishing cooperation, we rely on a number of task variants in which different skills must be learnt by the aerial robot before it can assist the ground robot in solving the task. In all task variants, six items are spread over an empty area of 550×350 cm, see Figure 5.1. The environment is unbounded, and the robots can thus roam away from the items and from each other. A simulation ends when all items are collected, or when the fixed time limit of 200 s has elapsed. Additional details are available in Appendix A.6. Each candidate solution (pair of ground/aerial robot controllers) is evaluated in ten independent simulations. The ground robot always starts in a random corner of the arena, facing a random direction, while the initial conditions of the aerial robot depend on the task variant, described below:

Fix-Tog: The aerial robot has no control over its altitude, but remains at the ideal sensing altitude (250 cm) throughout the whole simulation. The aerial robot starts directly above the ground robot.

Fix-Sep: Similar to *Fix-Tog*, but the aerial robot starts in a random location inside the arena, facing a random direction. This means that most of the time, the aerial and ground robots will not be within sensing range of each other at the beginning of a simulation.

TABLE 5.1: Configuration of the sensory inputs and actuators of the ground robot and aerial robot. See Figure 5.2 for an illustration.

Ground robot	**Aerial robot**
Sensory inputs	
$\mathbf{gi}_{1..4}$: 4 binary inputs that indicate the presence of items in the respective sector, within a 10 cm range. $\mathbf{ga}_{1..4}$: 4 binary inputs that indicate the presence of the aerial robot in the respective sector.[§]	$\mathbf{ai}_{1..6}$: 6 real-valued inputs that give the distance of the closest item in the respective sector.[§†] $\mathbf{ag}_{1..6}$: 6 real-valued inputs that give the horizontal distance to the ground robot in the respective sector.[§†] $\mathbf{ah}$: 1 input that gives the current altitude.[‡] $\mathbf{ac}_{d,\alpha}$: 2 inputs that give the distance and relative angle to the centre of the arena.
Actuators (maximum speed in parentheses)	
• Linear speed (15 cm/s)	• Front-back thrust (1 m/s) • Left-right thrust (1 m/s) • Up-down thrust (1 m/s)[‡]
• Turning speed (180°/s)	• Yaw rotation (90°/s)

[§] Based on the camera sensor. Horizontal range up to 144 cm, depending on the aerial robot's current altitude. [†] If no robot/object is present in the input's respective section, the maximum sensor value is returned. [‡] Not used in the *Fix-Tog* and *Fix-Sep* task variants.

Var-Tog: The aerial robot starts on the ground, next to the ground robot, and can freely move up and down.

Var-Mid: The aerial robot starts on the ground, but it is placed in a random location in the arena, up to a distance of 300 cm away from the ground robot.

Var-Sep: Similar to *Var-Mid*, but the aerial robot is placed in a random location in the arena, with no restrictions.

5.1.3 Evolutionary Setup

We use the same evolutionary setup as in the experiments in Section 3.5 and Chapter 4. The neural network controllers of each robot are evolved by NEAT (Stanley and Miikkulainen, 2002). The two coevolving populations use the same NEAT parameters, except for the number of input and output neurons due to the different number of sensors and actuators. The ground robot's neural network has 8 inputs and 2 outputs, while the aerial robot's network has 15 inputs and 4 outputs (14 inputs and 3 outputs in the Fix-* task variants). The remaining evolutionary algorithm parameters are listed in Appendix A.1.

The fitness score of a team, F_i, corresponds to the number of items that were successfully collected during the simulation trial. The team behaviour characterisation, used for novelty-driven coevolution and behavioural analysis, is described in Table 5.2. Each candidate solution (robot team) is evaluated in ten independent simulation trials. To obtain a more accurate estimate of the teams' quality, all the *best-of-generation* teams (Definition 7) were re-evaluated a-posteriori in 50 simulation trials. All the fitness plots presented in this chapter correspond to the results obtained in these post-evaluations.

TABLE 5.2: Team behaviour characterisation used in the aerial-ground foraging task. All features are normalised to $[0, 1]$.

Team behaviour characterisation
▷ Number of items collected
▷ Proportion of time the robots spent within the sensing range of one another
▷ Mean distance between the robots over the simulation time
▷ Mean distance of the robots to the closest item, averaged over the simulation

5.2 Standard Fitness-driven CCEA

We begin by studying the performance of the standard fitness-driven CCEA in the different task variants. The highest fitness scores achieved throughout evolution are depicted in Figure 5.3. Each evolutionary treatment was repeated in 30 independent evolutionary runs. The results show that there are clear performance differences in the five variants, with statistically significant differences between all setups (Mann-Whitney U test, $p < 0.001$) in the highest fitness scores achieved. The CCEA can consistently and quickly evolve high-fitness solutions for the *Fix-Tog* variant. In the other task variants, where the aerial robot needs to learn more complex skills before being able to cooperate, the CCEA's performance was significantly affected. Coevolution displayed the lowest performance in the *Var-Sep* variant, where cooperation is hardest to achieve.

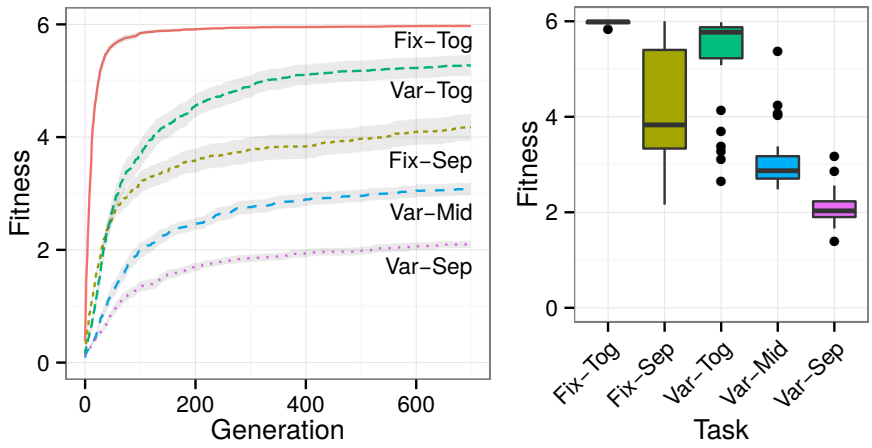


FIGURE 5.3: Fitness scores achieved by the standard *CCEA* in each of the task variants. Left: average of the highest fitness scores achieved at each generation. The grey areas depict the standard error. Right: boxplots of the highest scores achieved in each evolutionary run.

To determine the reasons behind evolutionary failure in the more challenging setups, we divided the evolutionary runs into two sets: the successful runs, which achieved a fitness score of at least 4; and the failed runs, which did not reach that mark. We then visually inspected some of the highest scoring solutions evolved in each of these runs, see Figure 5.4. These results reveal that the successful runs always relied on a high degree of cooperation to solve the task: the aerial robot was close to the ground robot most of the time. As it can be seen in the figure, the aerial robot first finds the ground robot (see the *Fix-Sep* and *Var-Mid*, successful runs) and then gets close to the items one at a time, while the ground robot is following it.

In the failed runs, however, we see a different scenario: the aerial robot displays a behaviour that almost seems to ignore the ground robot. It does not actively search

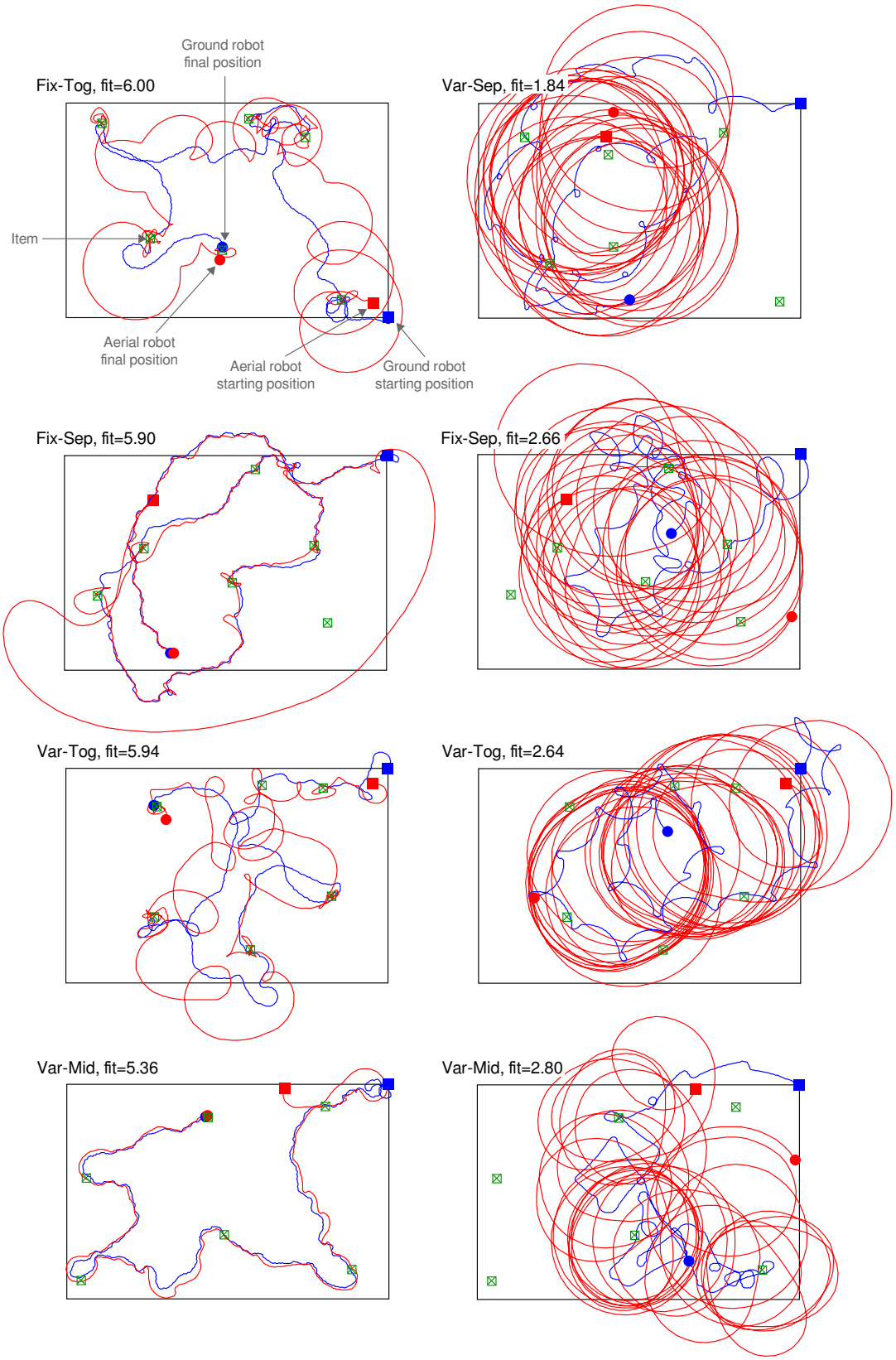


FIGURE 5.4: Examples of the highest-scoring solutions evolved in evolutionary runs of fitness-driven coevolution. Left column: best solutions evolved in successful runs. Right column: best solutions evolved in failed runs. The red line depicts the path of the aerial robot, and the blue line the path of the ground robot. The filled squares mark the initial positions, and the circles mark the final positions. The green squares with crosses mark the items. Videos available online at `https://doi.org/10.5281/zenodo.47066`.

for the items in the environment, but instead moves in circles over the arena. The ground robot uses the aerial robot's position to know where the arena is, but it has to search for the items alone using its very limited capabilities. The interaction between the two robots is therefore minimal or non-existent.

If the two populations fail to sustain a mutual development of skills, the individuals of one population can become over-adapted to the poor behaviours found in the other population, reaching an equilibrium from which it can be hard to escape. This degenerate dynamic corresponds to premature convergence to mediocre stable states, discussed in Section 2.4.2. In our aerial-ground foraging task, for instance, the flying robot can evolve a behaviour where it simply does circles inside the arena. The ground robot adapts to this behaviour, thus reaching an equilibrium state – a local optima in the collaboration space (Figure 5.4, failed runs).

To confirm that convergence to stable states was the culprit of the CCEA's low performance, we resorted to the analysis of the *best-of-generation* teams, as described in Section 3.3.1. For each of the *best-of-generation* teams, we measured the fitness (number of items collected) and the amount of time the robots spent within the sensing range of one another (*time within range*). This measure is directly related to the degree of cooperation between the robots, since the aerial robot cannot assist the ground robot if it is permanently outside its sensing range. The *average behaviour* of each generation was then calculated based on the mean values of fitness and *time within range* obtained in all *best-of-generation* teams (from the different evolutionary runs) of that generation. By plotting the average behaviour obtained throughout the generations of the evolutionary algorithm, it is possible to see to which types of solutions the evolutionary process is converging, see Figure 5.5.

The results from the successful runs show that, in all task setups, there is a close

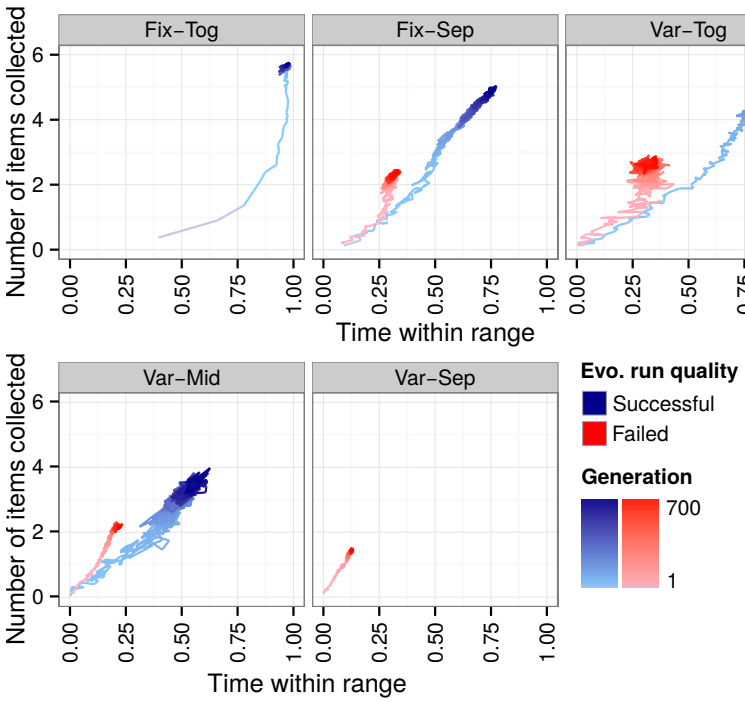


FIGURE 5.5: Average behaviour of the *best-of-generation* solutions evolved by the standard fitness-driven CCEA, grouped by successful (blue, highest fitness achieved ≥ 4) and failed runs (red, fitness < 4). The lighter colours denote the earlier generations. The *time within range* is the time the robots spent within the sensing range of each other.

relation between the amount of cooperation (*time within range*) and the fitness of the solutions. High-scoring solutions always display high levels of cooperation. In the *Fix-Tog* task, evolution quickly converges to (near-)optimal solutions without much exploration. In this task, there are high levels of cooperation right from the beginning, as the robots start near one another and the aerial robot has the optimal sensing altitude. In the other task variants, evolution takes significantly more generations to evolve solutions with high levels of cooperation.

In the failed runs, evolution does not reach solutions with high cooperation levels, which is consistent with the behaviours observed in Figure 5.4. The failed runs appear to be more biased towards the collection of items than robot cooperation, as evidenced by the higher numbers of items collected for the same levels of *time within range*. In the *Fix-Sep* and *Var-Tog* tasks, for instance, we can see that in the failed runs, evolution is trying to increase the number of items collected without increasing cooperation. In this foraging task it is, however, impossible to achieve high fitness scores without cooperation, and the results therefore indicate that evolution is trapped in a stable state from which it cannot escape.

5.3 Avoiding Premature Convergence

After identifying premature convergence to stable states as the cause of failure of the coevolutionary process, we evaluate the capability of novelty-driven cooperative coevolution (see Chapter 3) of overcoming it, and compare it with other competing techniques. The *Fix-Tog* task variant was not used in these experiments since the standard CCEA could always reach near-optimal solutions.

5.3.1 Methods

The strategy for solving this foraging task is relatively clear beforehand, as it is enforced by the morphological limitations of the robots: (i) the aerial robot must learn how to take-off and maintain an altitude that optimises the sensor coverage; (ii) the aerial robot must somehow find the ground robot; (iii) the robots must frequently be within sensing range of one another, otherwise they cannot cooperate; and (iv) the two robots must devise some cooperative strategy where the aerial robot leads the ground robot towards the items. This natural decomposition of the task allow us to compare novelty-driven cooperative coevolution with problem decomposition techniques that are based on the experimenter's knowledge, see Section 3.1: incremental evolution (Gomez and Miikkulainen, 1997), and multi-objective optimisation of subgoals (Mouret and Doncieux, 2008).

Standard Fitness-driven CCEA (Fit)

Standard cooperative coevolutionary algorithm, see Section 5.2, where the individuals are scored only according to the team's fitness score – the total number of items collected (F_i).

Novelty-driven Cooperative Coevolution (NS)

Novelty-driven coevolution uses the same implementation as the experiments in the previous chapters. We use the *NS-Team* technique (Algorithm 2), in which the fitness scores are combined with the novelty scores via Pareto multiobjective ranking. The parameters are listed in Appendix A.1.

Incremental Evolution (Inc)

Incremental evolution (Gomez and Miikkulainen, 1997) relies on the decomposition of the larger goal in sub-goals that are easier to achieve, thus overcoming bootstrap problems and premature convergence. We defined a sequence of sub-goals that try to bridge the gap between the number and complexity of skills each robot has to evolve. We encouraged the development of skills in the aerial robot, and the evolution of cooperation between the two robots, before trying to solve the ultimate objective of collecting items (F_i). At any moment in evolution, the chosen representative individual of each population was the individual that obtained the highest fitness score in the previous generation, according to the fitness function of the current sub-goal. We relied on the following sub-goals, in the order $(F_a) \rightarrow F_w \rightarrow F_i$:

1. Minimise the difference between the aerial robot's altitude (a_t) and the near-maximum sensing altitude (A, 240 cm) over the simulation trial (T steps). The goal is achieved when 20% of the individuals in a generation achieve a score of at least 0.9. This goal is not used in the *Fix-Sep* variant, as the aerial robot's altitude is fixed.
$$F_a = 1 - min\left(1, \sum_{t\in[1,T]} \frac{|a_t - A|}{T \cdot A}\right) \tag{5.1}$$

2. Maximise the time robots spend within the sensing range of one another (t_w). The goal is achieved when 20% of the individuals in a generation achieve a score of at least 0.7.
$$F_w = t_w/T \tag{5.2}$$

3. Maximise the number of items collected throughout the simulation run (F_i).

Multi-objective Evolutionary Algorithm (MOEA)

In this approach, the three objectives that are used in incremental evolution, are employed in a Pareto-based multi-objective evolutionary algorithm. The evolutionary process therefore tries to maximise the three objectives at the same time, instead of maximising them in a predefined sequence as in incremental evolution (Mouret and Doncieux, 2008). For the *Fix-Sep* variant, only two objectives (F_w and F_i) were used, since the altitude is fixed. The multi-objective optimisation followed the same implementation as the one used to combine novelty and fitness in *NS*, see Appendix A.1. The representative individual of each population is the individual that achieved the highest fitness score (F_i) in the previous generation, according to the results obtained in preliminary experiments. Other possibilities for the choice of the representative have been evaluated in preliminary experiments, see Appendix A.6.

5.3.2 Results

We conducted 30 independent evolutionary runs in each experimental setup (methods $\times$ task variants). The fitness scores achieved with each method over the evolutionary process are shown in Figure 5.6. Note that for all methods, the fitness score corresponds to the number of items collected (F_i), not necessarily the selection scores the individuals received during the evolutionary process. To understand how the studied methods can mitigate premature convergence and encourage cooperation, we performed an analysis of the *best-of-generation* solutions (similar to the one found in Section 5.2), see Figure 5.7.

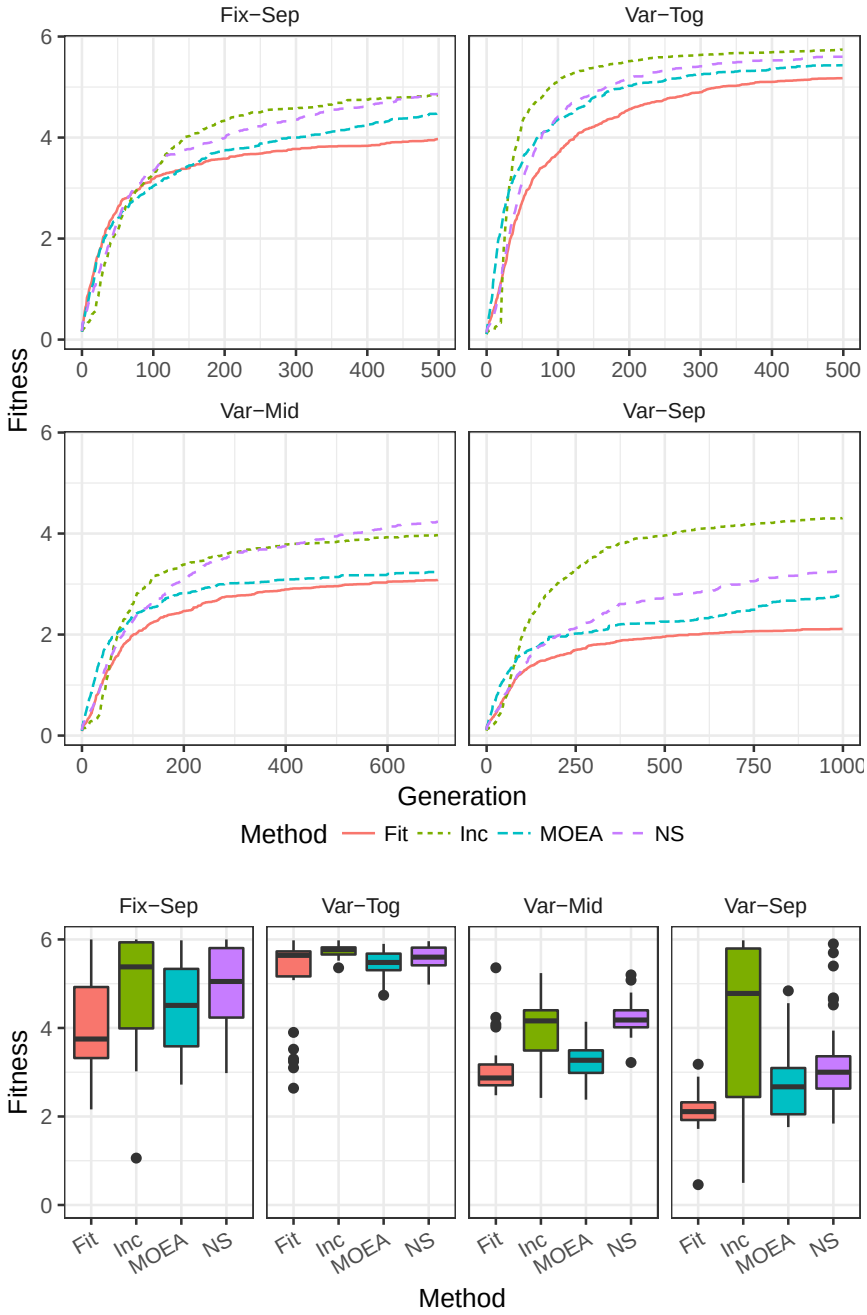


FIGURE 5.6: Top: average of the highest fitness scores achieved at each generation, for each task variant and method. Bottom: highest fitness scores achieved in the evolutionary runs. Fitness corresponds to the number of items collected (F_i).

Incremental Evolution (Inc)

Incremental evolution was on average the highest performing approach, and outperformed the standard CCEA in all task variants (Mann-Whitney U test, $p < 0.05$). The results in Figure 5.6 show that incremental evolution tends to reach high fitness scores in fewer generations than the other methods. Incremental evolution initially rewards the robots for staying within sensing range of one another. As the robots are essentially forced to cooperate before reaching the final stage, evolution is less likely to get stuck in a mediocre stable state where the robots do not cooperate when collecting the items.

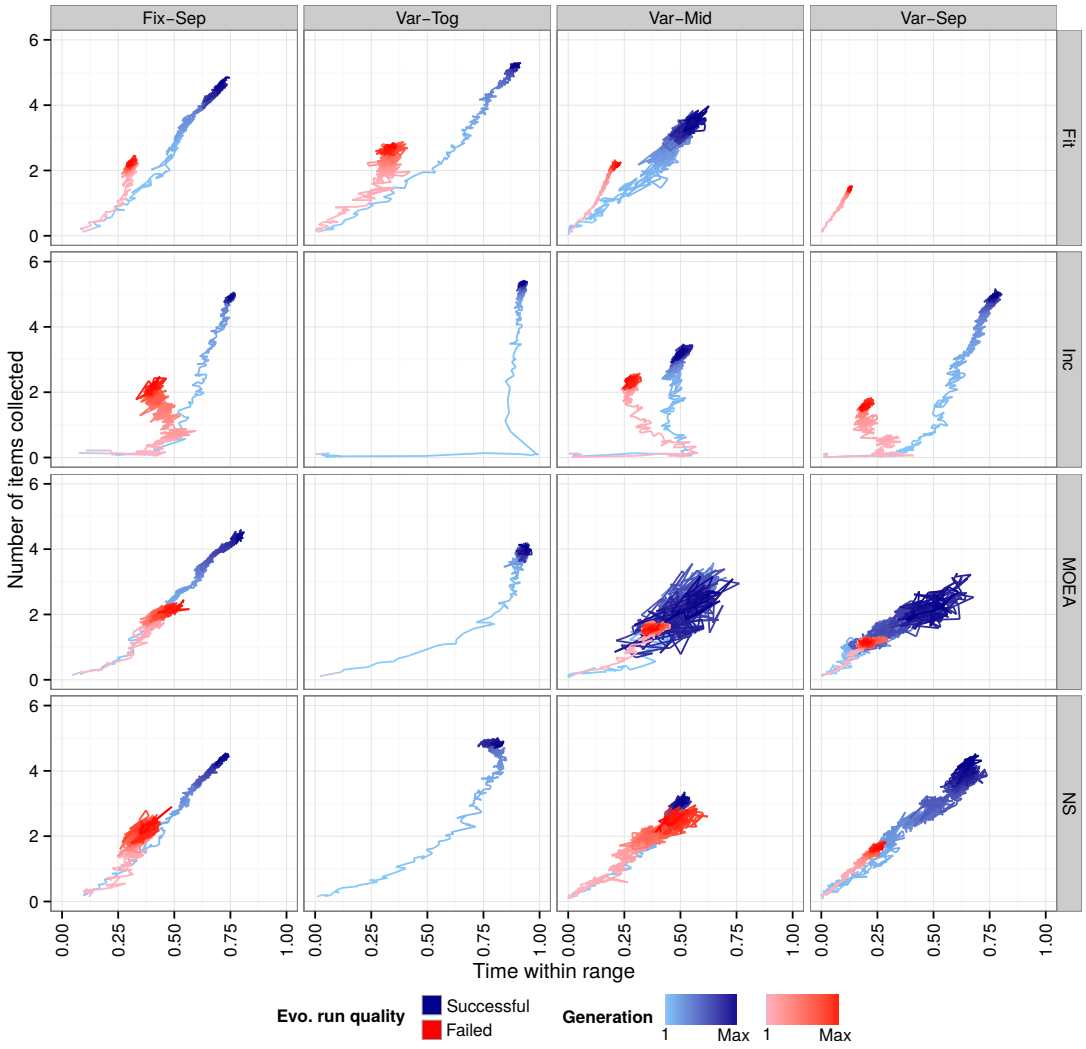


FIGURE 5.7: Average behaviour of the *best-of-generation* solutions evolved by each method, grouped by successful (blue, highest fitness achieved ≥ 4) and failed runs (red, fitness < 4). The lighter colours denote the earlier generations. The *time within range* is the time the robots spent within the sensing range of each other.

As Figure 5.7 shows, all evolutionary runs of *Inc* initially maximised the *time within range*, without increasing the number of items collected. When evolution reached the final stage, solutions started to maximise the number of items collected, and at this point two opposite scenarios could be observed: (i) evolution increased the fitness of solutions while increasing or maintaining the levels of *time within range*, leading to good solutions (successful runs); or (ii) evolution increased the fitness of solutions but the *time within range* decreased, ultimately leading to a mediocre stable state (failed runs).

Although the robots typically learn to find each other, they did not always evolve to successfully collect items. The effectiveness of incremental evolution depends on the task decomposition defined by the experimenter, both in terms of the definition of the sub-goals, as well as the transitions between those goals. A more fine-grained incremental configuration (with more sub-goals for instance) could potentially yield better performance, but it would also introduce additional biases in the evolutionary process.

Multi-objective Optimisation (MOEA)

Despite the potential advantages of the multi-objectivisation approach over incremental evolution (Mouret and Doncieux, 2008), our results failed to show such advantages. While MOEA achieved a similar performance to incremental evolution in the *Fix-Sep* task ($p = 0.10$), its performance was inferior in all the *Var-*$*$ tasks ($p < 0.001$). The MOEA approach also failed to improve over the standard fitness-driven coevolutionary algorithm in all tasks ($p > 0.05$) except *Var-Sep* ($p = 0.005$), see Figure 5.6.

The results in Figure 5.7 (MOEA row) reveal a relatively high variation of the average behaviour of the *best-of-generation* teams, especially in the *Var-Mid* and *Var-Sep* tasks, both in the number of items collected and the time within range. Although additional experiments would be needed to confirm the causes of MOEA's poor performance, these results suggest that the evolutionary algorithm was not able to strictly hold and advance the Pareto front across generations. This issue can potentially be explained by the fact that, unlike a traditional application of a MOEA, in a CCEA the objectives are not static: the objective scores given to any individual depend on the individuals with which it was evaluated.

Novelty-driven coevolution (NS)

Novelty-driven coevolution is substantially different from the approaches discussed above, as it rewards the exploration of the behaviour space, without introducing biases towards specific behaviours. Regarding the fitness scores achieved, *NS* significantly outperformed the standard CCEA in all variants ($p < 0.01$, Mann-Whitney U test), except in *Var-Tog* in which they achieved solutions of similar quality ($p = 0.54$). The performance of *NS* was only outperformed by *Inc* in the *Var-Sep* task ($p = 0.02$).

Another potential advantage of novelty-driven coevolution is the ability to discover a high diversity of solutions for a given problem. We confirm this advantage by analysing the behavioural diversity evolved in each run, using the average behavioural dispersion of all the evolved teams, as defined in Definition 9, Section 3.3.1. The results in Figure 5.8 show that novelty-driven coevolution exhibited a significantly higher degree of behaviour exploration than all other approaches in all task variants ($p < 0.05$), except in the *Var-Sep* task where the diversity evolved by novelty-driven coevolution matched that of incremental evolution ($p = 0.7$). These results highlight novelty search's ability to explore the behaviour space.

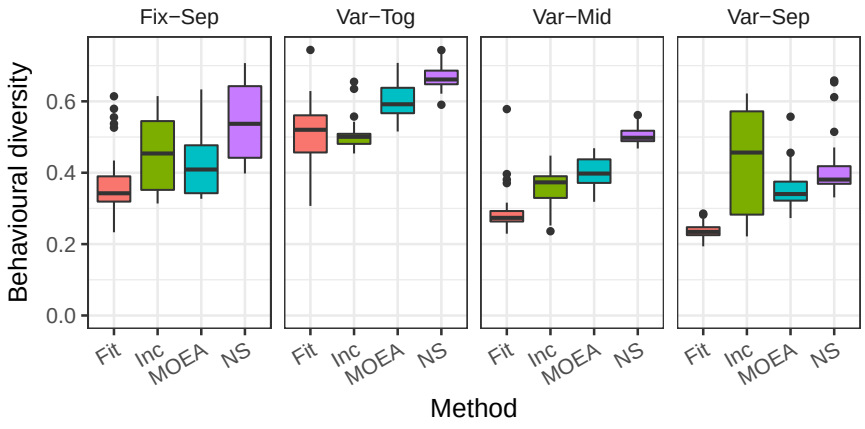


FIGURE 5.8: Behavioural diversity, calculated based on the mean difference between all individuals evolved over the course of each evolutionary run (Definition 9).

As stated above, in the *Var-Sep* task variant, *NS* was not always able to achieve successful solutions and a high-diversity of behaviours. In this aerial-ground foraging task, the team behaviour space can only be adequately explored if the two robots cooperate. If, for instance, the aerial robot does nothing at all, or simply flies away, the diversity of team behaviours that can be achieved is significantly limited. In these situations, the exploration of the behaviour space becomes impaired, and novelty-driven coevolution can thus fail to discover high-quality solutions.

5.4 Discussion

Coevolution with Heterogeneous Populations In the aerial-ground foraging task we use in this study, even the simplest task variant is associated with a significant heterogeneity in the robot team, with respect to sensor-effector capabilities, morphology, behaviours required for successful task execution, and complexity of neural network controllers. Nevertheless, cooperative coevolution consistently found (near-)optimal solutions for the simplest task variant. When the two robots can start cooperating from the very beginning of the evolutionary process, coevolution is able to sustain a mutual development of skills in the populations, leading them towards a (near-)optimal stable state.

In additional experiments (Gomes et al., 2016a), not reported in this document, we also showed that CCEAs can cope with an additional kind of heterogeneity: the use of different evolutionary algorithms in the different populations. We experimented coevolution with the NEAT algorithm on one population and a simple genetic algorithm evolving fixed topology networks on the other. Our results showed that the coevolutionary process is not significantly stifled by having different evolutionary algorithms, associated with significantly different learning speeds, operating simultaneously. We also showed that the two coevolving populations can effectively use different population sizes, which can potentially be used to optimise the coevolutionary process with respect to resource usage.

Limitations of the Standard CCEA Algorithm The experiments with the more challenging task variants revealed the limitations of the coevolution architecture when evolving heterogeneous agents. In the task variants where the aerial robot had to develop more complex skills before being able to cooperate, coevolution frequently failed. Our results showed that premature convergence to mediocre stable states was the main cause for this failure. If cooperation cannot be easily achieved in the beginning of the evolutionary process, evolution may gravitate towards solutions where the robots try to achieve the goal without relying on cooperation. The evolution of such behaviours can lead to stable states from which evolution cannot easily escape. The attraction to such stable states was relatively weak in the easier task variants, as most runs were successful. In the more challenging task variants, the attraction was stronger, and the standard coevolutionary algorithm always converged to such stable states and failed to evolve successful solutions.

Avoiding Premature Convergence Based on Manual Decomposition We tried to avoid convergence to mediocre stable states using problem decomposition techniques proposed in previous works. Incremental evolution (Gomez and Miikkulainen, 1997) was the most successful approach, significantly outperforming the

standard CCEA in all tasks. Incremental evolution is, however, associated with well-known limitations (Doncieux and Mouret, 2014), as a great deal of domain knowledge is required to design effective evolutionary stages. The need to manually define sub-goals, as well as appropriate transitions between these sub-goals, introduces strong biases in the evolutionary process (Doncieux and Mouret, 2014; Nelson et al., 2009), which counteracts the purpose of using evolutionary algorithms as black-box optimisers of agent controllers. Based on our intuition about the task, we defined three sub-goals, and although all stages were reached during evolution, many of the evolutionary runs still failed, especially in the more difficult task variants. This highlights the difficulty in properly configuring the stages and transitions in an incremental evolution scheme. We evaluated the multiobjectivisation of sub-goals as a way to overcome the difficulty in defining the order and transition of sub-goals (Mouret and Doncieux, 2008), but the method generally failed in this task, and was unable to improve over incremental evolution and even the standard CCEA.

Effectiveness of Novelty-driven Cooperative Coevolution Finally, we evaluated novelty-driven coevolution (Chapter 3), which rewarded individuals for displaying novel team behaviours. The performance of novelty search was similar to incremental evolution: it represented an improvement over the standard CCEA in all tasks. One advantage of novelty search over incremental evolution is that it relies less on the experimenter's knowledge and potential biases, thus leaving evolution more free to explore diverse solutions (Doncieux and Mouret, 2014). This advantage was confirmed in the comparison between the behavioural diversity generated by each algorithm: novelty search evolved a broader diversity of behaviours than any of the other algorithms tested. Novelty-driven coevolution, however, still failed frequently in the most challenging task variant. Our results suggest that these poor results are due to the difficulty of evolving any form of cooperation in this task variant – the robots simply finding one another in the environment can be challenging in itself – and without cooperation it is difficult to explore the behaviour space.

Towards Effective Coevolution with Heterogeneous Populations Our experiments showed that incorporating domain knowledge into the process (incremental evolution) or adopting a more open-ended evolutionary approach (novelty search) can mitigate premature convergence issues. Nevertheless, even when using these techniques, many of the evolutionary runs still failed in the most demanding task variants. One of the main issues was that the coevolutionary process often started converging to solutions where the robots did not cooperate with one another. The first step to avoid this problem is to rely, if possible, on a task setup where the multiple agents can start cooperating right from the beginning of evolution. Another possibility is to design the fitness function in such a way that it is impossible to improve the fitness of the team without relying on cooperation. That is, the fitness function can be tailored (Nelson et al., 2009) to avoid the mediocre stable states where productive cooperation does not exist.

5.5 Summary

In this chapter, we studied the challenges associated with coevolving behaviours for cooperative multirobot systems where there is a significant heterogeneity in the coevolving populations. Our experiments relied on a task where a highly capable aerial robot must assist a relatively simple ground robot in collecting items. We used

multiple task variants in which we varied the number and complexity of skills that the aerial robot had to develop before being able to cooperate with the ground robot. Our work contrasts with the vast majority of previous works that have only used cooperative coevolutionary algorithms with very similar populations and agents. To the best of our knowledge, the work presented in this chapter is the first to successfully demonstrate the evolution of controllers for a highly heterogeneous multirobot system.

Although these experiments were based on a single robotics domain, the main challenges that are addressed in this chapter are common to many multirobot tasks: how to foster synchronised learning, and how to encourage the evolution of cooperation in problems where it is not easily attainable. Overall, our results suggest that cooperative coevolution can work with an arbitrary level of heterogeneity in the populations, as long as the individuals from the different populations can establish a productive cooperation right from the beginning of the evolutionary process, thus leading to a mutual development of skills. When even the simplest forms of cooperation are hard to evolve, the coevolutionary process often converge to a mediocre stable state where cooperation is almost absent. We have shown that novelty-driven cooperative coevolution can alleviate this problem, and significantly improve the effectiveness of the coevolutionary process, but it was not a *silver bullet* – it was still affected by the difficulty in evolving cooperation in the most demanding task variant.

Chapter 6

Improving Scalability Through Dynamic Team Heterogeneity

In a typical CCEA application, as the ones presented in the previous chapters, the teams are *fully heterogeneous*, meaning there is a one-to-one mapping between populations and agents (Potter et al., 2001). Although ubiquitous, this design choice can limit the scalability of coevolutionary algorithms with respect to the number of agents, as discussed in Section 2.4.2. Similarly to a traditional CCEA, novelty-driven cooperative coevolution can, in theory, be equally affected by this limitation: (i) in case there are many agents in the team, it might be difficult to generate a novel team behaviour by modifying the controller of a single agent (credit assignment issue); and (ii) similar behaviours might be separately evolved for different agents, which is an inefficient usage of resources (redundant learning). A number of previous works have indeed shown that in large teams, many of the agents actually display very similar behaviours (Nitschke et al., 2010, 2012a). We have also shown (Gomes et al., 2014a) that, even with smaller teams of three agents, there is often an overlap between the behaviours evolved for different agents.

A natural solution for improving scalability is to assign each population to multiple agents, thereby forming homogeneous sub-teams inside the larger heterogeneous team (Luke et al., 1998; Panait and Luke, 2005a). This scheme does, however, require that a suitable *team composition* is known, i.e., how many different sub-teams there should be, and which agents should be assigned to each one. Such knowledge is rarely available, and it represents an additional bias introduced by the experimenter in the evolutionary process.

In this chapter, we propose Hyb-CCEA, an extension of the CCEA algorithm that takes into account the exploration of the behaviour space by each population to avoid the same behaviours being evolved in separate populations. Hyb-CCEA uses operators that allow the *merging* of separate populations, which decreases heterogeneity, and the *splitting* of populations, which increases heterogeneity. Hyb-CCEA thus departs from the fixed mapping between agents and populations, and the number of populations in the system becomes dynamic. Several agents in the team can be assigned to the same population, which has the potential to significantly improve the scalability of the evolutionary process. We first study Hyb-CCEA in an abstract domain, with the objective of understanding its capability to converge to suitable team compositions, its scalability in terms of the number of agents and problem complexity, the influence of its main parameters, and how Hyb-CCEA fares against competing approaches. We then show how Hyb-CCEA can be applied to concrete tasks without having to rely on predefined team compositions, using four simulated multirobot tasks: two multi-rover foraging tasks, and two robot soccer tasks.

It is important to clarify that Hyb-CCEA is not an extension of novelty-driven

cooperative coevolution (Chapter 3), but rather an extension of cooperative coevolutionary algorithms in general. Novelty-driven cooperative coevolution modifies the scores attributed to the population individuals, which is not affected by Hyb-CCEA. The two approaches are therefore algorithmically compatible with one another. We chose to evaluate the two approaches separately for clarity and generality of the results.

6.1 State of the Art

The reduction of heterogeneity in a multiagent system is the most direct way of improving the scalability of multiagent learning (Bongard, 2000; D'Ambrosio et al., 2010; Panait and Luke, 2005a). By reducing heterogeneity, the number of agent controllers that need to be learned decreases, thus reducing the search space (Luke et al., 1998). The evolution of partially heterogeneous multiagent systems is still relatively unexplored (D'Ambrosio et al., 2010; Waibel et al., 2009), with most of the previous works restricted to team learning (Lichocki et al., 2013), see Section 2.3. Team learning naturally facilitates the evolution of team compositions, since each genome encodes all the agent controllers and the team composition, thus allowing the optimisation of the team as a whole.

In the context of team learning, Hara, (1999) proposed a GP-based technique, *Automatically Defined Groups* (ADG), that automatically discovers the optimal number of groups and their compositions. Also based on genetic programming, Bongard, (2000) proposed the *Legion System*, where the genome encodes the composition of the team and one sub-tree for each behaviour class. It was shown that the amount of heterogeneity evolved by the system was dependent on the given problem domain, highlighting the importance of emergent team compositions. A radically different neuroevolution approach was proposed by D'Ambrosio and Stanley, (2008) and D'Ambrosio et al., (2010): all the agent controllers are indirectly encoded in a single genome using compositional pattern producing networks (CPPNs), which are evolved by the *HyperNEAT* algorithm (Stanley et al., 2009). HyperNEAT can exploit similarities in agents' policies, while at the same time allowing for variations. This means that there is no rigid concept of sub-team, but the agents can share part of their policies with one another.

In concurrent learning techniques, the emergence of team compositions is practically non-existent (Lichocki et al., 2013; Waibel et al., 2009). Typically, the team composition is configured a priori (most commonly, fully heterogeneous), and the CCEA only optimises the controllers of each agent type. To cope with large heterogeneous multirobot systems and facilitate the evolution of specialisations, Nitschke, (2008) proposed CONE (see Section 2.4.3), a cooperative coevolution approach that incorporates regulated breeding between different populations, each corresponding to a different agent. Crossover between different populations is allowed if the individuals of those populations share the same specialisation and have similar genotypes, thus mitigating the problem of reinvention.

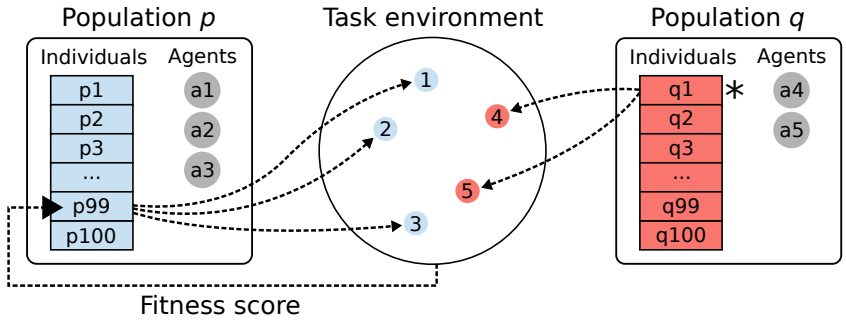
The teams evolved with CONE are nevertheless fully heterogeneous, as complete controllers are not shared by different agents. Hyb-CCEA, presented in this chapter, takes inspiration from CONE (Nitschke et al., 2010) in the sense that it relies on agent specialisations to regulate interactions between different populations. Hyb-CCEA, however, presents a number of theoretical advantages over CONE: (i) it allows for the emergence of genetically homogeneous sub-teams; (ii) it does not require the a priori specification of the possible specialisations; and (iii) CONE is an extension of

MESP, a neuroevolution algorithm bound to a specific neural network architecture. Hyb-CCEA, on the other hand, is compatible with any controller architecture and evolutionary algorithm, even beyond artificial neural networks.

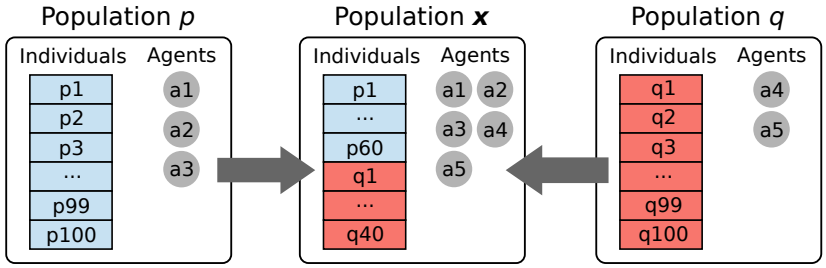
6.2 The Hyb-CCEA Approach

Hyb-CCEA departs from the one-to-one fixed mapping between agents and populations: we allow a population to be assigned to multiple agents, meaning that the controller encoded by each population individual can be copied to multiple agents in the team. A population in Hyb-CCEA can thus become responsible for the evolution of a genetically homogeneous sub-team, not just one specific agent.

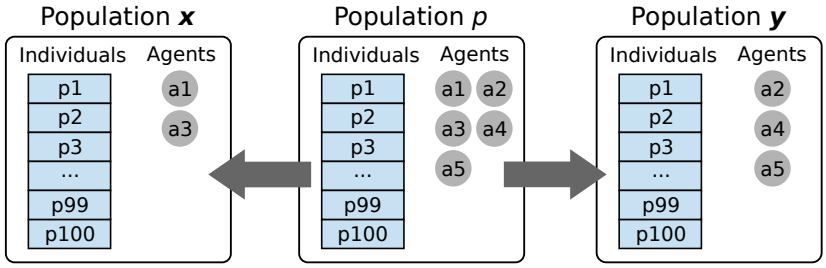
Each population is dynamically assigned to a subset of the agents in the team (Figure 6.1a), with every agent assigned to exactly one population. The rest of the



(a) Illustration of the **evaluation phase**. In this example, the individual *p99* of population *p* is being evaluated. The representative individual of *q* is *q1*. The population individuals (controllers) are assigned to the respective agents, and the fitness of the whole team is assigned to individual *p99*.



(b) In the **merge procedure**, the new population *x* replaces the two parent populations, *p* and *q*. The population *x* is formed by a subset of *p* and *q*'s individuals, and is assigned to the union of all the agents.



(c) In the **split procedure**, two new populations, *x* and *y*, replace the parent *p*. The populations *x* and *y* are copies of *p*, but each one is assigned to a disjoint set of agents.

FIGURE 6.1: Illustration of the main procedures in the Hyb-CCEA algorithm.

coevolutionary evaluation operates the same way as a traditional CCEA (Potter and De Jong, 2000): individuals are joined with representative individuals from the other populations for evaluation, and the individual being evaluated receives the fitness score that the team as a whole obtained. In this study, the representative individual of each population is the highest-fitness individual of the previous generation (Potter and De Jong, 2000; Wiegand et al., 2001).

The distinctive aspect of *Hyb-CCEA* is that it does not assume that the optimal number of sub-teams and their composition are known beforehand: we extend the CCEA so that the number and composition of the sub-teams is under evolutionary control. Different levels of heterogeneity can thus be explored by the evolutionary process. To this end, we propose: (i) a procedure for merging two populations, which creates a new population assigned to the agents of the two former populations, thus decreasing the heterogeneity of the system (Figure 6.1b); and (ii) a procedure for splitting a population, which creates two populations assigned to different sets of agents, thus increasing heterogeneity (Figure 6.1c). The following sections present the Hyb-CCEA algorithm in detail.

6.2.1 Evolutionary Process

The evolutionary process of Hyb-CCEA is described in Algorithm 5. It follows the general cooperative coevolution architecture (Potter and De Jong, 2000), with the main difference that the merge and split procedures are additionally executed every generation after the evaluation phase. Each population p in the system is a tuple composed of the population individuals ($\mathcal{I}_p$); the set of agents allocated to the population ($\mathcal{A}_p$); the number of generations passed since the creation of p (τ_p), also referred to as *age*; the maturation period (ℓ_p), which establishes the minimum lifetime of p; and the representative individual of the population (r_p), used during the evaluation of the other populations.

Algorithm 5 *Hyb-CCEA* algorithm.

1: $\mathcal{P} \leftarrow$ `InitialisePopulations`()
2: **for each** generation **do**
3: **for** $p \in \mathcal{P}$ **do**
4: **for each** individual $i \in \mathcal{I}_p$ **do**
5: $t_i \leftarrow \{i\} \cup \{r_q : q \in \mathcal{P} \wedge q \neq p\}$
6: $f_i, \beta_i \leftarrow$ `Evaluate`(t_i) assigning each individual of t to the respective set of agents
7: `AttemptMerge`$(\mathcal{P})$
8: `AttemptSplit`$(\mathcal{P})$
9: **for** $p \in \mathcal{P}$ **do**
10: $r_p \leftarrow$ individual $i \in \mathcal{I}_p$ with maximum f_i
11: **if** $\tau_p > 0$ **then**
12: $\mathcal{I}_p \leftarrow$ `Breed`$(\mathcal{I}_p)$ based on the fitness scores f
13: $\tau_p \leftarrow \tau_p + 1$

6.2.2 Initialisation

The `InitialisePopulations` procedure is described in Algorithm 6. With a total of n agents in the team, the algorithm can be initialised with any number of populations ranging from 1 to n. This means that the team can start fully homogeneous,

with only one population initially and all agents assigned to it; it can start fully heterogeneous, with one population assigned to each agent; or it can start with a partially heterogeneous configuration. Besides the agent allocation, each population is initialised with a random maturation period ℓ_p, drawn uniformly from 1 to the maturation limit T_L, and the representative individual r_p is initially chosen randomly among the population individuals $\mathcal{I}_p$, as typically done in CCEAs (Potter and De Jong, 2000).

Algorithm 6 `InitialisePopulations` procedure.

1: Let A the set of all agents
2: Let Ψ be the initial population-agent allocations, such that Ψ is a partition of A: $\bigsqcup_{\psi \in \Psi} \psi = A$
3: $\mathcal{P} \leftarrow \emptyset$
4: **for** $\psi \in \Psi$ **do**
5: $\quad \mathcal{A}_p \leftarrow \psi$ ▷ Assigned agents
6: $\quad \mathcal{I}_p \leftarrow \texttt{RandomIndividuals}(S)$ ▷ Initial population of size S
7: $\quad \tau_p \leftarrow 1$ ▷ Age
8: $\quad \ell_p \leftarrow \texttt{Random}(1, T_L)$ ▷ Maturation period
9: $\quad r_p \leftarrow \mathcal{I}_p[\texttt{Random}(1, S)]$ ▷ Representative individual
10: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{p\}$
11: **return** $\mathcal{P}$

6.2.3 Population Merge

Previous works have shown that cross-breeding between agents that share similar specialisations (Nitschke et al., 2010), or belong to the same sub-team (Luke and Spector, 1996), can be beneficial for the emergence of specialisations. Hyb-CCEA goes beyond this concept: if two separate populations are evolving agents with similar behaviours, they can be merged into a single population, and those agents thus become genetically homogeneous (Figure 6.1b). To identify behavioural similarities between agents, we rely on an *agent behaviour characterisation* (similar to the behaviour characterisations used for *NS-Ind*, Section 3.2.2), which can be provided by the experimenter or automatically derived, as discussed in Section 2.5.4. This characterisation is a real-valued vector $\beta_{i,a}$ composed of features that describe the behaviour of the agent a, obtained in the evaluation of the individual i (step 6 of Algorithm 5).

The merge procedure is described in Algorithm 7. Only populations with an age (τ_p) greater than their maturation period (ℓ_p) are eligible for merging (step 1). We first obtain sets of agent behaviours that are representative of each eligible population (steps 5 and 7). The behaviour set $\mathcal{B}_p$ of a population p is obtained by aggregating the agent behaviours recorded during the evaluation of the individuals of p. Since the objective is to identify the behaviour space region to which a population is converging, we only consider the `Elite` of the population, i.e., the fraction T_E of the population with the highest fitness scores.

We then measure the distance between these sets of behaviours (step 8). The distance between two behaviour sets is given by the silhouette index (or coefficient) (Rousseeuw, 1987), a clustering index that measures both the cohesion and separation of a given clustering. By considering that each behaviour set is a different cluster, the silhouette index thus measures the overlap between the two sets. The

Algorithm 7 `AttemptMerge` procedure.

1: $\mathcal{P}' \leftarrow \{p \in \mathcal{P} : \tau_p > \ell_p\}$
2: **if** $|\mathcal{P}'| < 2$ **then**
3: **return**
4: **for** $p \in \mathcal{P}'$ **do**
5: $\mathcal{B}_p \leftarrow \{\beta_{i,a} : i \in \texttt{Elite}(\mathcal{I}_p, T_E) \wedge a \in \mathcal{A}_p\}$
6: **for** $q \in \mathcal{P}' \wedge q \neq p$ **do**
7: $\mathcal{B}_q \leftarrow \{\beta_{i,a} : i \in \texttt{Elite}(\mathcal{I}_q, T_E) \wedge a \in \mathcal{A}_q\}$
8: $\mathcal{C}_{p,q} \leftarrow \texttt{SilhouetteIndex}(\mathcal{B}_p \cup \mathcal{B}_q)$
9: $p, q \leftarrow \arg\min_{p,q} \mathcal{C}_{p,q}$
10: **if** $\mathcal{C}_{p,q} \leq T_M$ **then**
11: $\mathcal{A}_x \leftarrow \mathcal{A}_p \cup \mathcal{A}_q$
12: $\mathcal{I}_x \leftarrow \texttt{Elite}\left(\mathcal{I}_p, \frac{|\mathcal{A}_p|}{|\mathcal{A}_x|}\right) \cup \texttt{Elite}\left(\mathcal{I}_q, \frac{|\mathcal{A}_q|}{|\mathcal{A}_x|}\right)$
13: $\tau_x \leftarrow 0$
14: $\ell_x \leftarrow \texttt{Random}(1, T_L)$
15: $\mathcal{P} \leftarrow (\mathcal{P} \setminus \{p, q\}) \cup \{x\}$

silhouette index will tend towards 1 if the two behaviour sets are cohesive and well separated, and will be close to 0 if there is a significant overlap between them. Negative silhouette indexes mean that the behaviours are on average more similar to the other behaviour set than their own. Let $a(i)$ be the average distance of the behaviour vector i to all other elements within the same behaviour set, and $b(i)$ be the average distance of i to all the elements in another behaviour set. The silhouette index can be calculated by averaging the silhouette values of all the behaviour vectors:

$$\texttt{SilhouetteIndex}(\mathcal{X}) = \frac{1}{|\mathcal{X}|} \sum_{i \in \mathcal{X}} \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{6.1}$$

Finally, the two most similar populations are merged if the distance between them is inferior to the merge threshold T_M (steps 9 and 10). The new population replaces the two parent populations, and is comprised by the fittest individuals from the respective populations. The number of individuals that are drawn from each population is proportional to the number of agents assigned to that population. The new population is assigned a random maturation period, between 1 and T_L.

6.2.4 Population Split

The *split procedure* increases the heterogeneity of the system by dividing a homogeneous sub-team into two new sub-teams: two clones of the population are created, and each one is assigned to a disjoint set of agents, see Figure 6.1c and Algorithm 8. The populations with an age τ_p above their maturation period ℓ_p and that are assigned to more than one agent are eligible to be split (step 1). Among these populations (if any), the population with the highest age is split (step 4). The set of agents of the parent population is randomly partitioned into two subsets, and each of the child populations is assigned to one of those subsets. The individuals of the two resulting populations are exact copies of the individuals of the parent population.

Contrary to the merging procedure, agent behaviour characterisations cannot be used to regulate splits, as agents assigned to the same population are genetically homogeneous, and will likely display very similar behaviours. Stochastic splits are an

Algorithm 8 `AttemptSplit` procedure.

1: $\mathcal{P}' \leftarrow \{p \in \mathcal{P} : \tau_p > \ell_p \wedge |\mathcal{A}_p| > 1\}$
2: **if** $\mathcal{P}' = \emptyset$ **then**
3: **return**
4: $p \leftarrow \arg\max_{p \in \mathcal{P}'} \tau_p$
5: $\mathcal{A}_x, \mathcal{A}_y \leftarrow$ Randomly partition $\mathcal{A}_p$ such that: $\mathcal{A}_x \sqcup \mathcal{A}_y = \mathcal{A}_p$
6: $\mathcal{I}_x, \mathcal{I}_y \leftarrow \mathcal{I}_p$
7: $\tau_x, \tau_y \leftarrow 0$
8: $\ell_x, \ell_y \leftarrow \texttt{Random}(1, T_L)$
9: $\mathcal{P} \leftarrow (\mathcal{P} \setminus \{p\}) \cup \{x, y\}$

effective solution because unfavourable splits can later be reverted by the *merge procedure*. The two new populations are assigned the same random maturation period (step 8), so that there is the chance of merging them later on, before being further split. If the two recently split populations did not diverge to different behaviours after the maturation period, the behaviour distance between them will be relatively small, and they will therefore be merged again.

6.3 Comprehensive Evaluation in an Abstract Domain

To systematically study the Hyb-CCEA approach, we propose an abstract domain – the *coverage task* – in which the objective is to evolve a set of *agents* ($\subset \mathbb{R}^N$) that cover a randomly generated set of *targets* ($\subset \mathbb{R}^N$). A similar domain has previously been used in (Lichocki et al., 2013), but focused only on the evolution of team compositions, where each agent and specialisation is unidimensional. The coverage task allows us to study both the evolution of agent controllers and team compositions in a wide diversity of problem instances, with different complexities and requirements. Although not directly applicable to real-world problems, we believe that the proposed domain contains the essential properties of more realistic multiagent domains, as explained below.

6.3.1 Problem Definition

Let T be a set of targets, randomly generated at the beginning of the evolutionary run, and A be a set of agents under evolution, such that:

$$T = \{\mathbf{t}_1, \cdots, \mathbf{t}_M : \mathbf{t}_i \in [0,1]^N\} \subset \mathbb{R}^N \tag{6.2}$$

$$A = \{\mathbf{a}_1, \cdots, \mathbf{a}_M : \mathbf{a}_i \in [0,1]^N\} \subset \mathbb{R}^N \tag{6.3}$$

The fitness function scores solutions based on the similarity between the set of agents A and the set of targets T in Euclidean space, such that there is an arbitrary one-to-one correspondence between agents and targets. The fitness function is defined in Algorithm 9. The global optimum is unique and trivial: $A = T$, with the order of the elements being irrelevant. For the Hyb-CCEA algorithm, we defined the agent behaviour characterisation as the distance of the agent to each of the targets:

$$\beta(\mathbf{a}) = \{d(\mathbf{a}, \mathbf{t}) : \mathbf{t} \in T\} \tag{6.4}$$

Algorithm 9 Fitness function for the coverage problem.

Let A be the set of agents under evolution, T the set of targets, and N the number of dimensions.
$\mu \leftarrow 0$
while $|T| > 0$ **do**
  $\mathbf{a}, \mathbf{t} \leftarrow \underset{\mathbf{a} \in A, \mathbf{t} \in T}{\arg\min}\ d(\mathbf{a}, \mathbf{t})$
  $\mu \leftarrow \mu + d(\mathbf{a}, \mathbf{t})$
  $A \leftarrow A \setminus \{\mathbf{a}\}$
  $T \leftarrow T \setminus \{\mathbf{t}\}$
return $1 - \frac{\mu}{\sqrt{N}}$

This problem domain is an abstraction of a multiagent task, where the targets (T) are specialisations that must be found by the evolutionary process. Similarly to a cooperative multiagent task, there is no absolute notion of individual fitness, as the agents (A) are not allocated to any target a priori, i.e., the fitness of one agent also depends on the position of the other coevolving agents. The populations can only rely on the global fitness for the evaluation of individuals. By allowing the set of targets to contain duplicate elements, multiple agents might need to converge to the same target, which is also common in multiagent tasks (Nitschke et al., 2010). By varying the number of *unique targets*, we can control the optimal degree of heterogeneity in the agent team, ranging from fully homogeneous (all targets are identical), to fully heterogeneous (all targets are different), to anything in between (only some targets are identical). Additionally, the complexity of the problem can easily be configured by varying the number of dimensions (N).

6.3.2 Evolutionary Setup

We use a standard genetic algorithm to evolve the agents. Each new chromosome is generated either by crossover or mutation, with equal probability. Individuals are selected using tournament selection, genes are mutated individually with a fixed probability, and we use one-point crossover. The elite of each population passes directly on to the next generation. See Appendix A.7 for parameter values. Every evolutionary treatment is repeated in 30 independent evolutionary runs, in which the run number (from 1 to 30) is used as the random seed to generate different sets of targets T. This ensures that different evolutionary treatments use the same sets of targets. The default parameters of the Hyb-CCEA algorithm, used in the experiments in this chapter unless indicated otherwise, are listed in Table 6.1.

TABLE 6.1: Default parameters for the Hyb-CCEA algorithm.

Parameter		Value
Merge threshold	T_M	0.2
Maturation limit	T_L	20
Behaviour similarity elite	T_E	0.2

6.3.3 Comparison with Competing Approaches

We begin by comparing Hyb-CCEA with other competing approaches, namely two standard CCEA algorithms, where each population is isolated, and a CCEA algorithm where individuals can migrate between different populations:

- **CCEA-H:** Fully heterogeneous CCEA – one population per agent.
- **CCEA-PH:** CCEA where each population is assigned to a predefined set of agents, such that the optimal team composition for the given task is achieved. The number of populations is thus equal to the number of unique targets.
- **C-Exch:** Similar to *CCEA-H*, but every generation, each population replaces 25% of their individuals with individuals imported from other behaviourally similar populations (if any). The behavioural similarity is computed the same way as in *Hyb-CCEA*, and the similarity threshold is the same as the merge threshold.

For these experiments, the number of agents A is 10, the number of unique targets is 1, 3, 5, and 10, and the number of dimensions N is 30. The Hyb-CCEA algorithm was configured with a fully heterogeneous initialisation, and the parameters listed in Table 6.1. The results in Figure 6.2 (top) show that *CCEA-H*, *CCEA-PH* and *Hyb-CCEA* can achieve high fitness scores (> 0.995) across all problem instances.

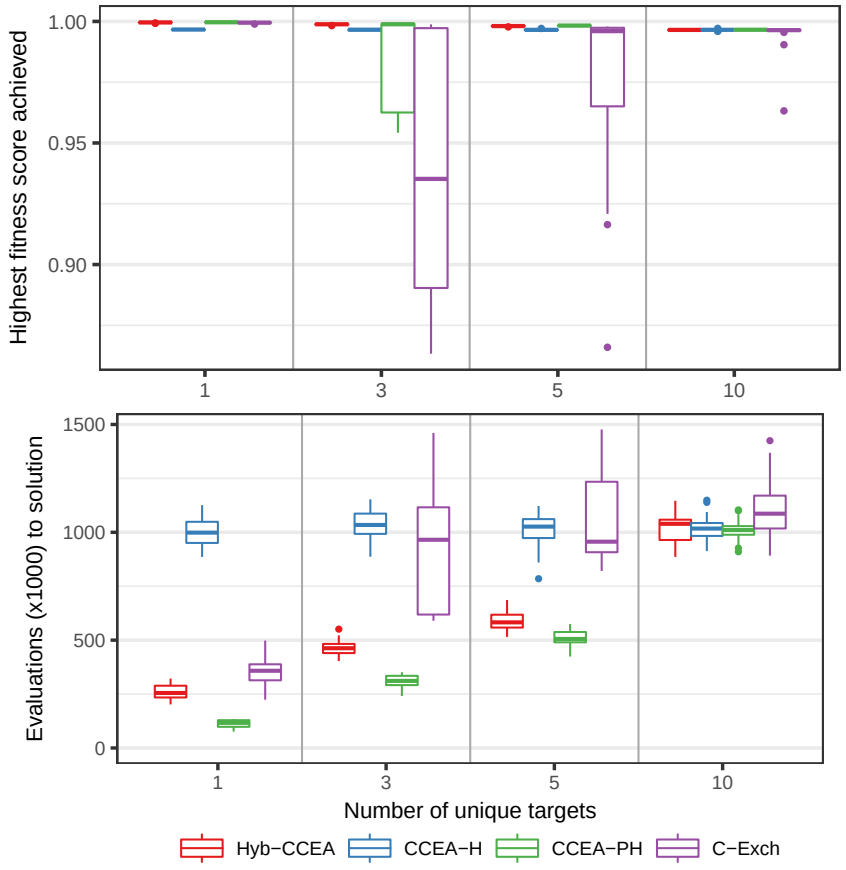


FIGURE 6.2: Top: highest fitness scores achieved with the different methods, for different problem instances. Bottom: number of evaluations needed on average to achieve a fitness level of 0.995.

C-Exch was only able to consistently achieve high fitness scores in the problem instances with 1 or 10 unique targets.

To distinguish between the methods that achieved high fitness scores, we analysed how many evaluations are needed to achieve the fitness threshold of 0.995, see Figure 6.2 (bottom). The results show that when the number of unique targets is less than the number of agents (10), Hyb-CCEA needs significantly fewer evaluations than the fully heterogeneous CCEA (*CCEA-H*) to reach the fitness threshold (Mann-Whitney, $p < 0.001$). When the number of unique targets is the same as the number of agents, and a fully heterogeneous team is thus necessary, Hyb-CCEA needs a similar number of evaluations as *CCEA-H* to reach the fitness threshold ($p = 0.63$). Comparing Hyb-CCEA with the CCEA with the optimal team allocation (*CCEA-PH*), Hyb-CCEA needs more evaluations to reach the same fitness level ($p < 0.01$), except when a fully heterogeneous team is needed (10 unique targets). This difference is, however, relatively small in magnitude, and remains constant across the problem instances.

Overall, these results suggest that Hyb-CCEA can approximate the performance of a CCEA with the optimal team composition. Hyb-CCEA, however, does not require the optimal team composition to be known a priori. The results also show that there is a low overhead of using Hyb-CCEA for tasks in which partial heterogeneity is not beneficial.

6.3.4 Scalability with Problem Complexity

In this section, we assess the robustness of Hyb-CCEA to problems with varying degrees of complexity. To this end, we vary the number of dimensions (N), and the number of unique targets, while keeping the number of agents (M) fixed to 10. We configured Hyb-CCEA with a fully heterogeneous initialisation and the default parameters listed in Table 6.1.

The results in Figure 6.3 (left) show that Hyb-CCEA was able to achieve near-optimal solutions in all problem instances. The fitness scores achieved by Hyb-CCEA decrease slightly and predictably as the number of dimensions (N) increases.

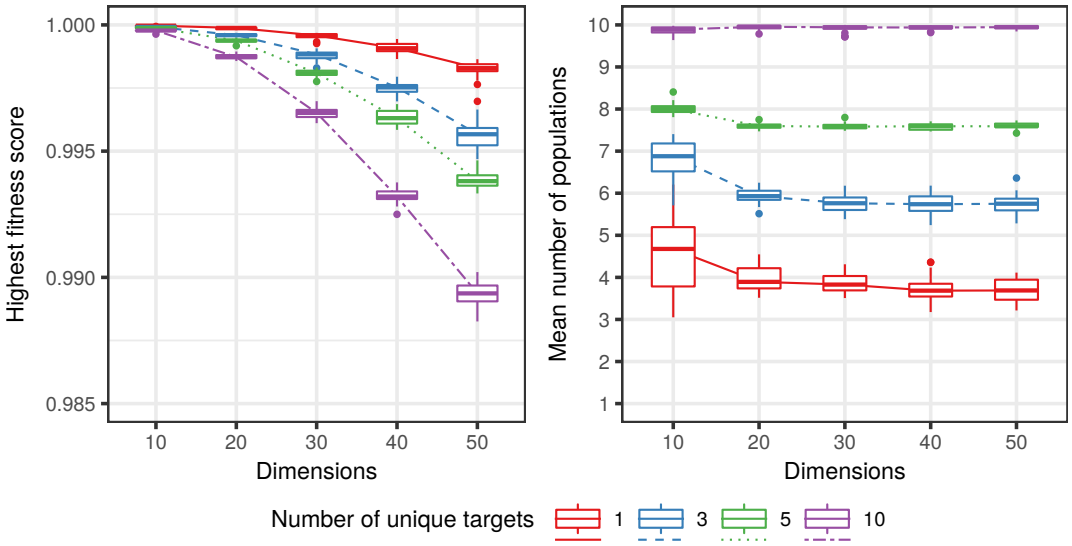


FIGURE 6.3: Left: highest fitness scores achieved by Hyb-CCEA in each problem instance (number of dimensions × number of unique targets). Right: mean number of populations during the evolutionary process, for each problem instance.

To understand how Hyb-CCEA adapts to the different problem instances, we analysed the mean number of populations throughout the evolutionary runs, see Figure 6.3 (right). The results reveal that the mean number of populations is dependent on the number of unique targets, but it is mostly independent from the number of dimensions. This shows that Hyb-CCEA is capable of converging to suitable team compositions, regardless of the problem complexity. In the problem instances with less than 10 unique targets, the mean number of populations is always slightly higher than the optimal team composition because: (i) Hyb-CCEA starts fully heterogeneous (10 populations), and needs some generations to converge; and (ii) stochastic splits continue to happen throughout the evolutionary process, thus temporarily increasing the number of populations.

6.3.5 Scalability with Respect to Team Size

We assessed the capability of Hyb-CCEA to scale to larger numbers of agents and populations. The number of dimensions was set to $N = 30$, and we varied the number of agents M from 2 to 50. The number of unique targets was also varied, ranging from 1 to M. As in the previous experiments, we configured Hyb-CCEA with a fully heterogeneous initialisation and the parameters listed in Table 6.1. The evaluation budget was varied proportionally to the number of agents: 300K, 750K, 1.5M, 3M, and 7.5M for M = 2, 5, 10, 30, 50, respectively.

The Hyb-CCEA algorithm was able to consistently find near-optimal solutions for all combinations of number of agents and number of unique targets – the mean of the highest fitness scores achieved is above the threshold of 0.995 in all problem instances. In Figure 6.4, we show the mean number of populations used by Hyb-CCEA in each problem instance. The mean number of populations is close to the number of unique targets, for every number of agents. There is a high linear correlation (Pearson's $r = 0.98$) between the number of unique targets and the mean number of populations. As discussed in the previous section, the absolute difference between the mean number of populations and the number of unique targets (i.e., the optimal

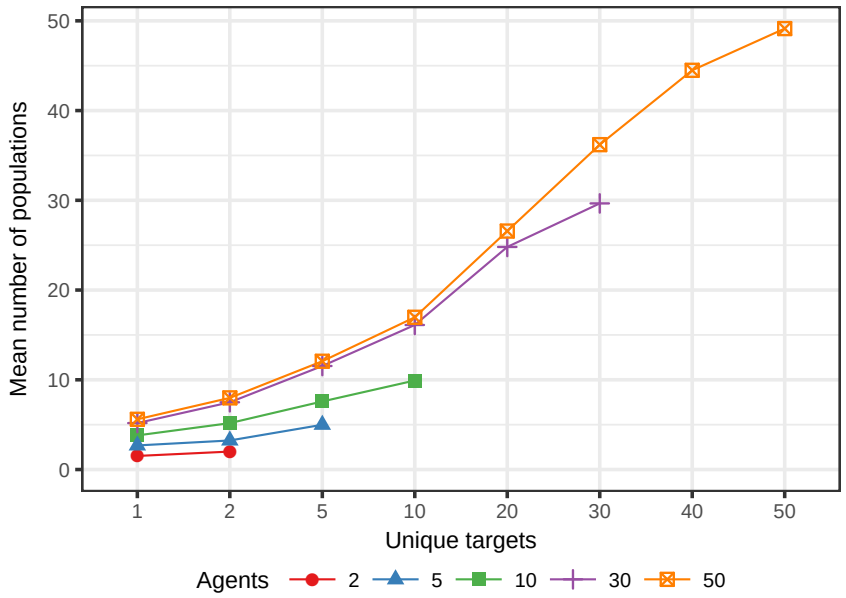


FIGURE 6.4: Mean number of populations in Hyb-CCEA for each problem instance (number of unique targets × number of agents).

number of populations) is explained by the time the algorithm takes to converge, and the population splits that continue to occur throughout evolution. These experiments confirm the potential of Hyb-CCEA to evolve controllers for teams composed of high numbers of agents.

6.3.6 Initial Team Composition

In this section, we assess if and how the provided initial team composition (see Section 6.2.2) biases the outcome of the Hyb-CCEA evolutionary process. For these experiments, the number of dimensions is set to $N = 30$, the number of agents set to $M = 10$, the number of unique targets is variable from 1 to M, and we vary the starting conditions, ranging from fully homogeneous (1 population) to fully heterogeneous (10 populations).

The results in Figure 6.5 show that, regardless of the initial configuration, Hyb-CCEA tends to converge to approximately the same number of populations. We did not observe significantly different outcomes for the different starting conditions, regarding the mean number of populations in the final 500K evaluations of the evolutionary runs, for any of the problem instances (Kruskall-Wallis test, $p > 0.05$). The relatively quick convergence of the algorithm to the same team composition translates to a similar performance regardless of the initial composition – for each problem instance, there were also no significant differences in the fitness scores achieved (Kruskall-Wallis, $p > 0.05$). These results suggest that the initial team composition does not have a significant impact on the effectiveness of Hyb-CCEA.

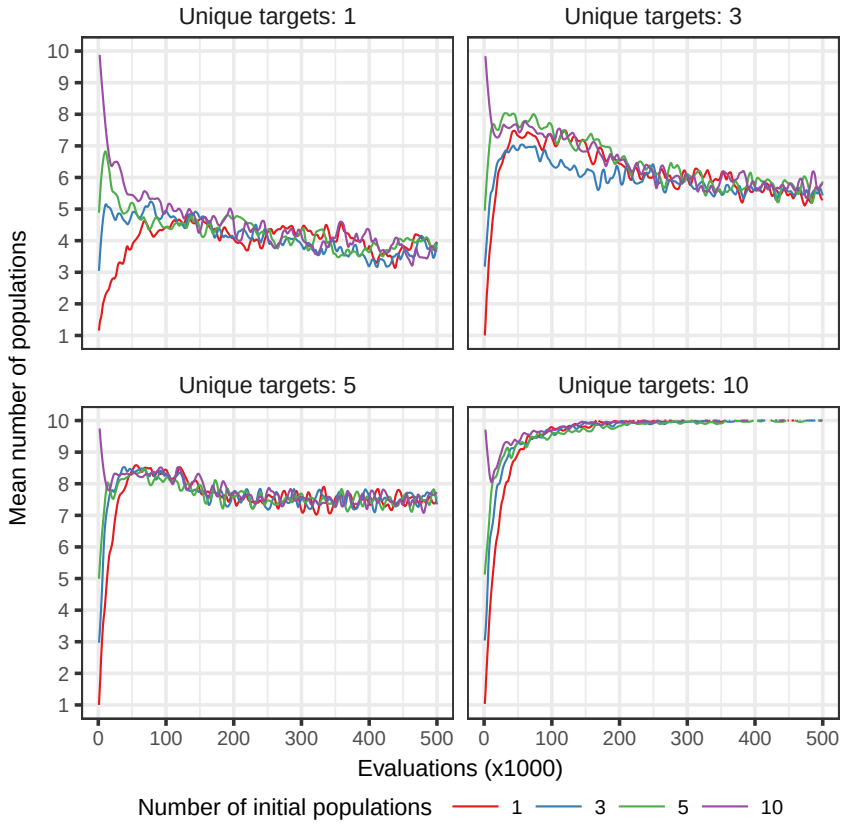


FIGURE 6.5: Mean number of populations in Hyb-CCEA throughout the evolutionary process, for the different problem instances and different initialisation conditions. Only the first 500K evaluations of a total of 1.5M are shown.

6.3.7 Merge Threshold and Maturation Limit

In this section, we study the two main parameters of Hyb-CCEA: the merge threshold (T_M) and the maturation limit (T_L), see Sections 6.2.3 and 6.2.4. To assess the impact of these two parameters, we used the same problem instances as in Section 6.3.3, where we varied the number of unique targets, while keeping the number of agents (M) fixed to 10 and set the number of dimensions N to 30. For each problem instance, we tested multiple combinations of T_M and T_L. In Figure 6.6, we show the highest fitness scores achieved by each configuration, in Figure 6.7, we show the mean number of evaluations needed to reach solutions, and in Figure 6.8, we show the mean number of populations during the evolutionary process.

In the problem instance that requires a homogeneous team (1 unique target), the number of evaluations tends to decrease with an increase of the merge threshold, regardless of the maturation limit (average Spearman's correlation $r = -0.96$). Increasing the merge threshold always facilitates the merging of populations (Figure 6.8), which naturally favours a problem instance where a fully homogeneous teams is preferable. When (partially) heterogeneous teams are required, there is a relatively wide range of T_M values for which the algorithm performs well. Beyond certain values, however, performance can drastically decrease (Figure 6.6). This is explained by the fact that beyond certain threshold values, Hyb-CCEA starts merging populations that should not be merged, as evidenced by the rapid decay in the mean number of populations, thus impeding the evolution of good solutions. In general, decreasing the merge threshold too much can cancel the advantages of Hyb-CCEA, but increasing it too much can prevent the algorithm to reach optimal solutions.

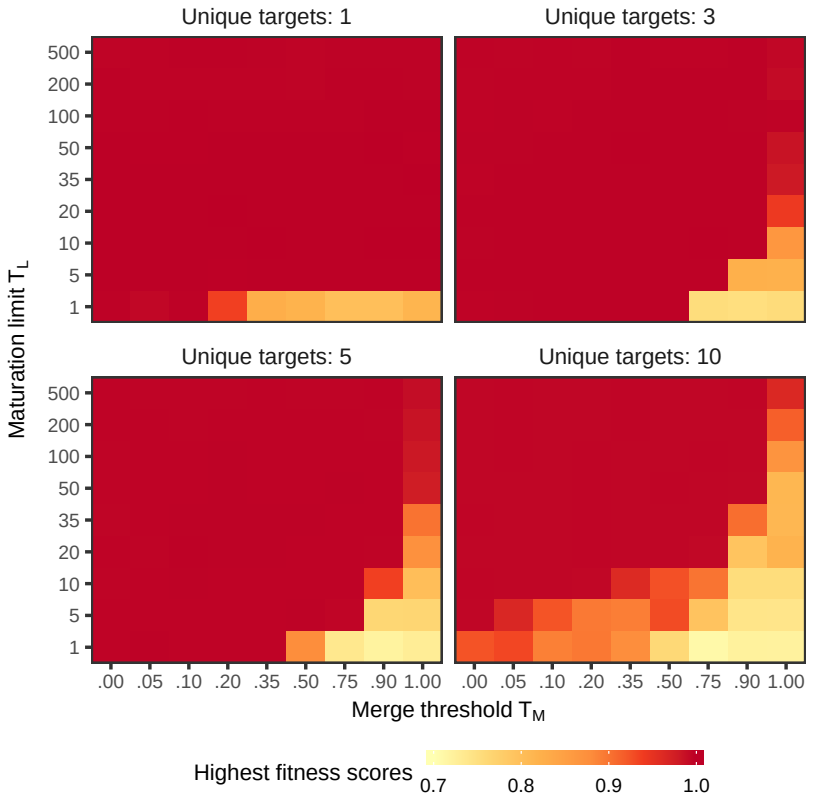


FIGURE 6.6: Highest fitness scores achieved by the evolutionary runs, averaged over 30 runs for each configuration of Hyb-CCEA (higher is better).

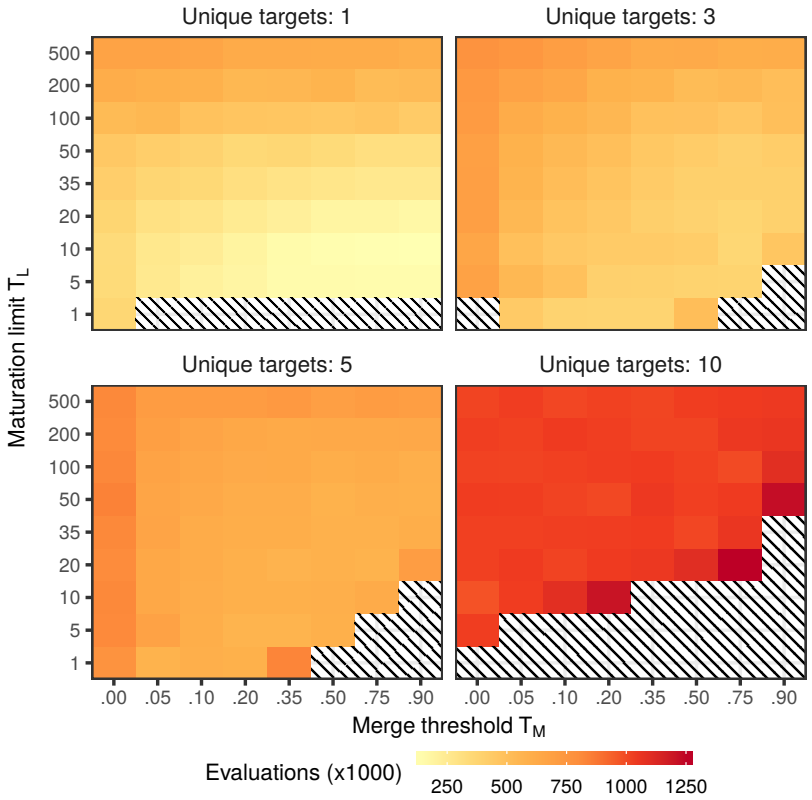


FIGURE 6.7: Average number of evaluations needed to achieve a fitness level of 0.995, for each configuration of Hyb-CCEA (lower is better). The textured areas correspond to configurations that could not achieve that fitness level in at least 20 out of 30 runs.

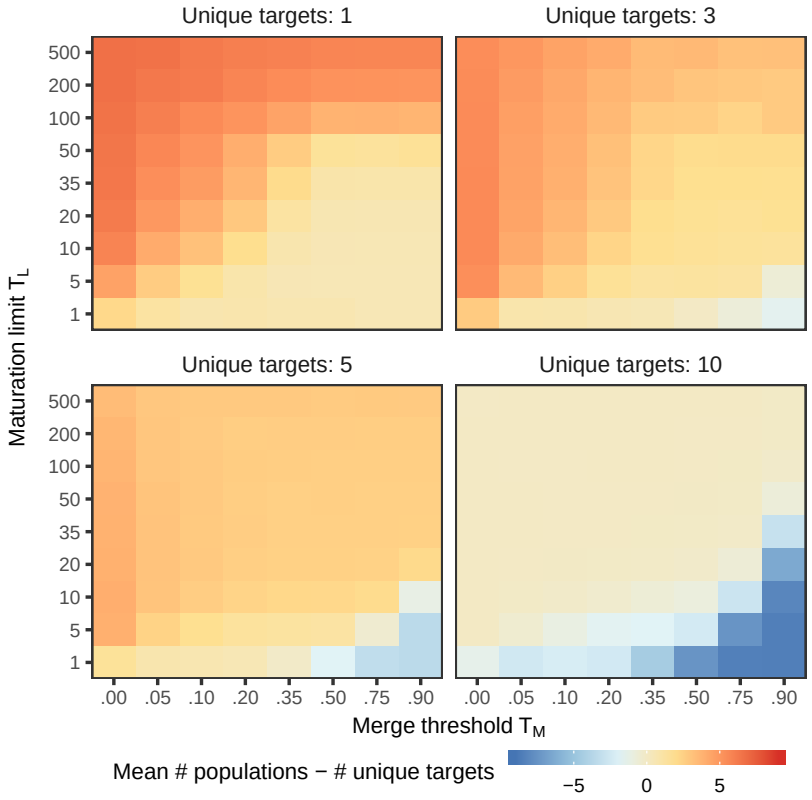


FIGURE 6.8: Average difference between the mean number of populations during the evolutionary runs and the number of unique targets, for each configuration.

Regarding the maturation limit, the results show that very short maturation limits (≤ 10 generations) tend to be prejudicial to Hyb-CCEA's performance, resulting in many configurations where evolution was unable to reach solutions (see textured areas in Figure 6.7). Short maturation periods might not provide enough time for the populations to converge to different specialisations, which is particularly evident by the negative impact of short maturation limits in the problem instance where a heterogeneous team is needed (10 unique targets). This negative impact becomes increasingly more pronounced as the merge threshold is increased, as the populations need to become more specialised in order to remain separated. Allowing very long maturation periods, on the other hand, can lead to a slower convergence of the algorithm to the optimal team composition.

Overall, our results show that Hyb-CCEA is moderately robust to variations in the maturation limit and merge threshold. For every problem instance, there is a wide range of parameter values for which Hyb-CCEA successfully finds near-optimal solutions. Moreover, there was a range of configurations that was highly successful across all problem instances: $T_M \in [0.20, 0.50] \wedge T_L \in [20, 50]$.

6.4 Validation in Simulated Multirobot Systems

In this section, we study how Hyb-CCEA can be applied to multirobot systems. As described in Section 6.2.3, Hyb-CCEA requires an agent behaviour characterisation for the merge procedure. It is an essential part of the algorithm that allows behaviourally similar populations to be identified and ultimately merged. We study two different approaches for the definition of the behaviour characterisation: generic characterisations that do not rely on the experimenter's knowledge; and task-specific characterisations that must be provided by the experimenter. An overview of behaviour characterisations, including the generic characterisations developed in the context of this thesis, is provided in Section 2.5.4. We validate Hyb-CCEA using four simulated multirobot tasks: two variants of cooperative foraging and two variants of robot soccer.

6.4.1 Generic Agent Behaviour Characterisation

In this chapter, we adopt one of the simplest measures that we proposed in (Gomes and Christensen, 2013): the *sampled average state*. This measure obtains a behaviour characterisation by averaging the sensory-effector values of the robots over time windows evenly spaced across the evaluation time. For simplicity, we consider a single time window (the whole evaluation time), meaning that this measure requires no parameter tuning. The behaviour characterisation of an agent a is given by:

$$\beta(a) = \langle \overline{v_{1_a}}, \cdots, \overline{v_{n_a}} \rangle \ , \tag{6.5}$$

where $\overline{v_{i_a}}$ is the mean normalised value of the i-th sensor/effector of agent a over the evaluation time.

6.4.2 Multi-rover Foraging Task

The multi-rover foraging task requires a team of agents to find and collect items in a bounded environment. We extended the cooperative foraging task used in Chapter 3 to support more agents and to require different specialisations, making it similar to the *extended multi-rover task* used by Nitschke et al., (2010). In the task used in

this chapter, a fixed number of items is randomly distributed in a bounded arena, see Figure 6.9. The agents start in the centre of the arena, and have the objective of capturing as much items as possible in the given time limit. The items can have different types, each type associated with one different sensor resolution. Each agent can only activate one sensor resolution at a time, and it can only sense and collect the item types that match the currently active sensor resolution. For simplicity, an item is collected when an agent passes over it.

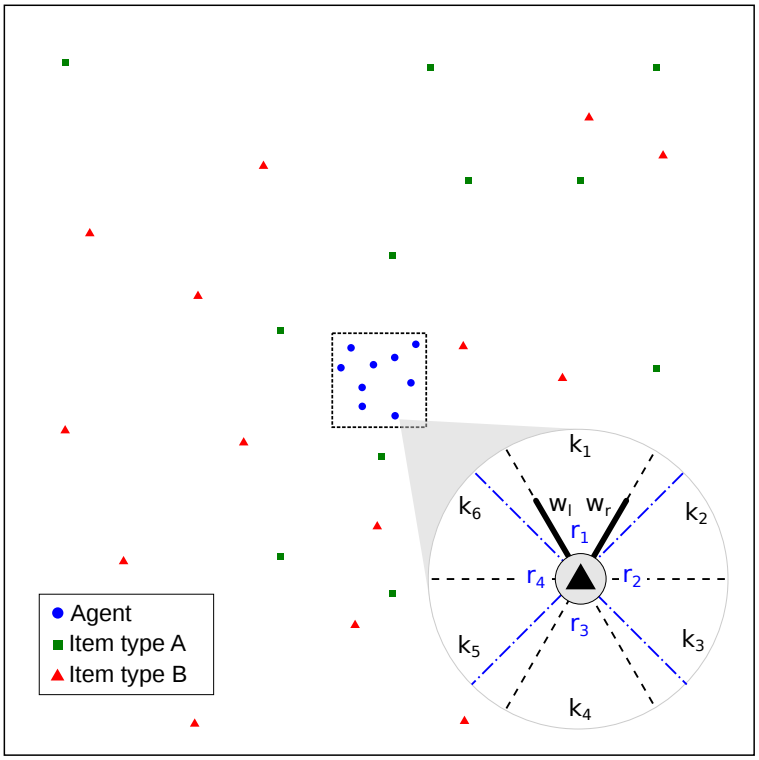


FIGURE 6.9: Initial conditions of the multi-rover foraging task with two item types. Each agent has two short-range sensors to detect walls (w_l, w_r), 6 sensors with unlimited range to detect the closest item (matching the current sensor resolution) in the respective circular sector ($k_{1..6}$), four sensors to detect the closest agent in the respective sector ($r_{1..4}$), and one sensor that returns the current sensor resolution of the nearest agent (n_R). Each agent has two actuators that control the linear and turning speed, and one actuator that dictates the sensor resolution.

The domain is particularly interesting for this study because the optimal behavioural specialisations depend on the item types that exist in the environment (Nitschke et al., 2010). If there is a single item type, for instance, it should be advantageous to have a fully homogeneous team. When there are more item types, the optimal team composition is less obvious – behavioural specialisation can be beneficial, with certain groups of agents specialised in searching for specific item types, but a viable alternative can be to have generalist agents that frequently switch between different sensor resolutions. We experiment with two environments: one with two different item types; and other with five different item types. The full list of task parameters can be found in Appendix A.8.

The fitness score assigned to an individual is the number of items collected by the team, averaged over 5 independent simulation runs, with randomised initial conditions (locations of items and initial positions of agents). The task-specific agent behaviour characterisation is a vector of length 6 or 12 (depending on the task variant), described in Table 6.2. The generic behaviour characterisation is a vector of

length 16 (the total number of sensors and actuators).

TABLE 6.2: Agent behaviour characterisations for a given agent a, for the multi-rover foraging and soccer tasks. All features are normalised to $[0, 1]$.

Multi-rover foraging task	Soccer task
▷ Number of items of each type collected by a	▷ Mean distance of a to the opponent's goal
▷ Amount of time each sensor resolution was active	▷ Mean distance of a to the ball
▷ Mean distance of a to the closest item	▷ Whether a scored a goal or not
▷ Mean distance of a to the closest agent	

6.4.3 Soccer Task

The soccer task is based on the RoboCup 2D simulated league[1] and on the SoccerBots domain of the TeamBots simulator.[2] Simulated soccer tasks remain a considerable challenge regarding the control of the teams (Barrett and Stone, 2015), and they have been used in numerous previous evolutionary robotics studies (Didi and Nitschke, 2016; Fehérvári and Elmenreich, 2010; Gomes et al., 2014a; Nelson et al., 2004; Scheepers and Engelbrecht, 2014). In the task used in this chapter, the robots' passing and kicking mechanics are abstracted as we did in (Gomes et al., 2014a), since we focus on team strategy and not on fine sensorimotor control. When the robot is over the ball, it simply chooses the direction and power of the pass, and the ball moves accordingly.

In our task, two teams of five players each play against one other. The controllers of one team of robots is under evolution, while the agents of the opponent team have a manually programmed controller. The manually programmed controller is an improved version of a controller available in TeamBots – *AIKHomoG*, see details in Appendix A.9. This controller uses dynamic role assignment for strategy and potential fields for choosing the movement direction. It has shown to be one of the best-performing teams available in TeamBots (Kose et al., 2005; Ramani et al., 2008). We experiment with two task variants, with different parameters for the preprogrammed controller:

- **Soccer-80%**: the opponent (pre-programmed) agents can only move and kick the ball at 80% the maximum speed.
- **Soccer-100%**: the opponent agents can move and kick at the same speed as the agents under evolution.

Each soccer game starts with the agents randomly placed in their respective halves, see Figure 6.10. For the team that starts with the ball, one of the agents (always the same one) is initially located close to the ball. The game ends when a goal is scored, when the time limit is reached, or when the ball gets stuck. A total of ten games are played, alternating which team starts with the ball. The task parameters are listed in Appendix A.9. The fitness of the evolving team is obtained by

[1] `http://wiki.robocup.org/wiki/Soccer_Simulation_League`
[2] `http://www.teambots.org/`

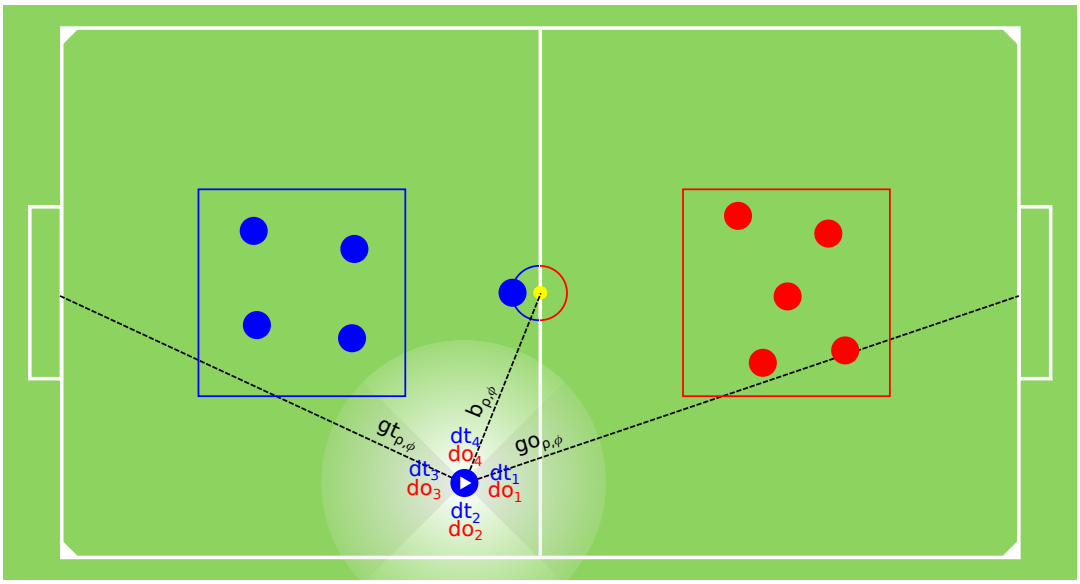


FIGURE 6.10: Initial conditions of the soccer task, with the left team starting. An additional agent is shown to illustrate the sensory configuration of the agents under evolution. Agents have four sensors with unlimited range that return the distance to the closest teammate in the respective circular sector ($dt_{1..4}$); four sensors that operate similarly to $dt_{1..4}$, but for the opponents ($do_{1..4}$); and six sensory inputs that return the distance and relative orientation to the ball ($b_{\rho,\phi}$), the own goal ($gt_{\rho,\phi}$), and the opponents' goal ($go_{\rho,\phi}$). The four actuators control the movement speed and direction, and the kicking power and direction.

averaging F_s over the ten games:

$$F_s = \begin{cases} 1 & \text{scored} \\ 0 & \text{otherwise} \end{cases} + \frac{1 - \frac{1}{T \cdot D} \sum_{t=1}^{T} \text{dist}(\mathbf{b}_t, \mathbf{g})}{100} \tag{6.6}$$

where T is the game length, D is the field's diagonal, $\mathbf{b}_t$ is the position of the ball at instant t, and $\mathbf{g}$ is the centre of the opponent team's goal. The second component of F_s rewards the team for keeping the ball close to the opponent's goal, thus encouraging good defense and offense strategies simultaneously. The second component is used to bootstrap evolution, as it is highly unlikely that randomly generated controllers can score any goals. The manually defined task-specific agent behaviour characterisation is a vector of length 3, described in Table 6.2. The generic behaviour characterisation is a vector of length 18 (the total number of sensors and actuators).

6.4.4 Evolutionary Setup

The controllers for both tasks were evolved using the NEAT algorithm (Stanley and Miikkulainen, 2002). The parameters of the NEAT algorithm were the same for both tasks, and are listed in Appendix A.1. It should be noted that in the case of Hyb-CCEA, the co-evolving NEAT populations all use the same *innovation counter*. This is done in order to allow the *merging* of separate populations, avoiding problems with the crossover between individuals that could have colliding innovation numbers. We used the default parameter values for Hyb-CCEA, listed in Table 6.1.

6.4.5 Results

For each task variant, we compared three approaches:

- **Hyb-CCEA-GC:** Hyb-CCEA algorithm, with fully homogeneous initialisation, using the generic behaviour characterisation (Section 6.4.1).
- **Hyb-CCEA-TS:** Hyb-CCEA algorithm, with fully homogeneous initialisation, using the task-specific agent behaviour characterisations (see Table 6.2).
- **CCEA:** Fully heterogeneous CCEA.

In Figure 6.11, we show the fitness scores achieved by each method in the different tasks. In all tasks, both variants of Hyb-CCEA outperformed CCEA, achieving significantly higher fitness scores (Mann-Whitney, $p < 0.001$), and requiring significantly fewer evaluations to reach similar fitness levels. The visual inspection of the highest-scoring solutions for each task (see videos online[3]) revealed that the best teams clearly relied on partial heterogeneity, with the different homogeneous sub-teams displaying different and complementary specialisations. In the multi-rover foraging task, each sub-team specialised in foraging specific item types, sometimes two item types at once by frequently switching the sensor resolution. In the soccer

[3]Videos of some of the solutions evolved with Hyb-CCEA and the standard CCEA, for both the soccer and multi-rover tasks, are available at `https://doi.org/10.5281/zenodo.292797`.

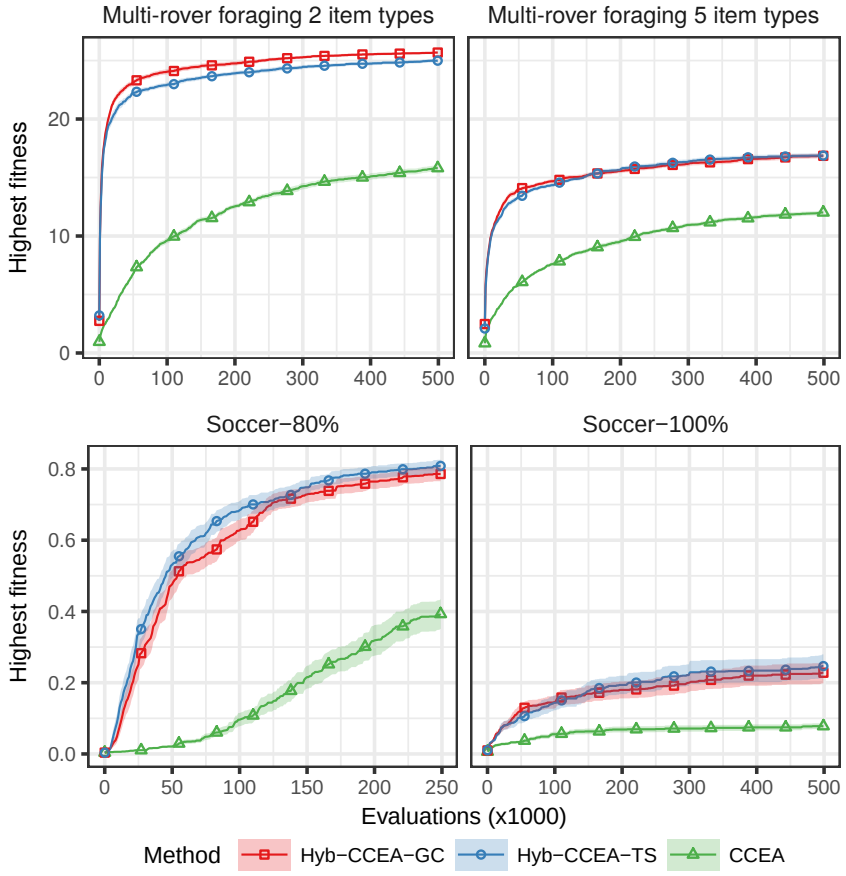


FIGURE 6.11: Highest fitness scores achieved on average at each number of evaluations, for the three methods and four task variants. The shaded area shows the standard error.

task, the teams were typically divided into two or three separate roles, with two or three agents actively trying to score a goal, and the other agents staying behind in the field to either defend the goal or stop opponents' attacks.

Regarding the fitness scores achieved by the two variants of Hyb-CCEA, we did not find statistically significant differences between the generic and task-specific characterisations except in the multi-rover task with two item types, where the generic characterisation (Hyb-CCEA-GC) yielded slightly higher fitness scores than the task-specific characterisation (Hyb-CCEA-TS) ($p < 0.001$). By considering the mean number of populations throughout the evolutionary process, see Figure 6.12, we can see that both Hyb-CCEA variants converge to similar numbers of populations. There were, however, significant differences between the two characterisations in all tasks, regarding the mean number of populations ($p < 0.01$, Mann-Whitney). These experiments show that the generic characterisations are able to effectively capture the behavioural features necessary to distinguish the different agents, and can perform as well as, or even better, than task-specific behaviour characterisations.

In the multi-rover foraging task, the variant with five item types naturally benefits from more agent specialisations (higher degree of team heterogeneity) than the variant with two item types. The results of Hyb-CCEA are consistent with this difference, with the algorithm converging to higher number of populations in the task variant with five item types. In the soccer task, Hyb-CCEA also converged to partially heterogeneous teams: in both task variants, the mean number of populations

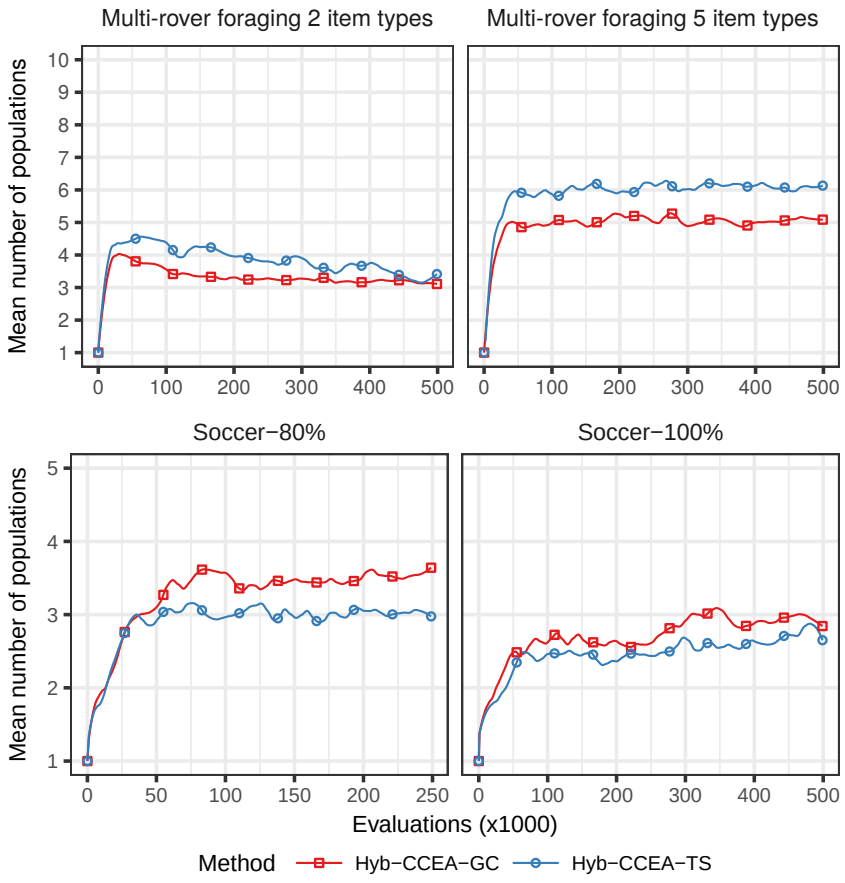


FIGURE 6.12: Mean number of populations throughout the evolutionary process, for each of the four tasks and the two variants of Hyb-CCEA.

is approximately half the number of agents. The results for the multirobot tasks are thus in line with the results for the coverage task presented in Section 6.3, and confirm that Hyb-CCEA often finds a suitable team composition and agent controllers for the tasks considered.

6.5 Discussion

Finding suitable team compositions We began by evaluating the properties of Hyb-CCEA with an abstract domain for which the global optimum is known, as well as the optimal team composition. The initial experiments with different instances of the coverage task revealed that Hyb-CCEA converges to the optimal team composition for the given problem instance, ranging from fully homogeneous teams, to fully heterogeneous, and compositions in between. The experiments showed that the algorithm scales well with the problem complexity, converging to the same team composition regardless of the dimensionality of the problem. We additionally showed that the algorithm scales with the number of agents, as the algorithm performed successfully in problem instances ranging from 2 to 50 agents.

Improving scalability of CCEAs Using the coverage task, we compared Hyb-CCEA with a traditional fully heterogeneous CCEA, and with a CCEA where the number of populations was constant, but there was exchange of individuals among behaviourally similar populations. Hyb-CCEA significantly outperformed the competing approaches across different problem instances. Even when the task required a fully heterogeneous team, Hyb-CCEA performed as well as the fully heterogeneous CCEA, and the advantage of Hyb-CCEA over the heterogeneous CCEA increased as more homogeneous teams were required. Generally, Hyb-CCEA and the traditional CCEA achieved similar fitness scores, but Hyb-CCEA required significantly fewer evaluations to achieve those scores.

We then successfully applied Hyb-CCEA to more complex multirobot tasks, where the optimal team composition was not known beforehand. These experiments revealed that the advantages of Hyb-CCEA go beyond reducing the number of evaluations needed to achieve solutions. In all four tasks used, Hyb-CCEA achieved significantly higher fitness scores than the traditional CCEA, besides reaching the same fitness levels with significantly fewer evaluations. Hyb-CCEA has the potential to mitigate the problem of credit assignment (Agogino and Tumer, 2008), since modifications in a single individual (which can be assigned to multiple agents) can have a greater impact in the team's performance. This can help overcome the issues of stagnation and premature convergence caused by vanishing fitness gradients.

Besides these multirobot tasks, we also evaluated an initial version of Hyb-CCEA in a herding task (Gomes et al., 2015b). We used six task variants, with five to ten agents. The results are consistent with the experiments in this chapter: Hyb-CCEA could achieve higher fitness scores than CCEA, and could achieve the same fitness levels using significantly fewer evaluations. The performance differences were more expressive in the task variants with more agents, thus supporting the potential of Hyb-CCEA to improve the scalability of the coevolutionary process.

Parameter sensitivity and generalisation Hyb-CCEA requires three main parameters to be provided by the experimenter: the merge threshold, the maturation limit, and the initial team composition. Our results showed that the algorithm is moderately robust to variations of all these parameters: (i) the algorithm converged to the

same team composition regardless of the initial composition; (ii) the merge threshold is dimensionless, independent from the task and the chosen behaviour characterisation; and (iii) there is relatively wide range of combinations of merge threshold and maturation limit that yielded successful results across all the tested problem instances.

Our experiments also showed that generic (task agnostic) agent behaviour characterisations can be successfully employed in Hyb-CCEA, thus freeing the experimenter from having to provide a behaviour characterisation tailored to the given task. The performance of the algorithm with generic characterisations was similar or better than the performance obtained with task-specific behaviour characterisations.

Limitations We identified two main issues that should be further studied in future work. First, due to the periodic and stochastic population splits, the number of populations in the algorithm is on average above the number of populations needed to optimally solve the task. This could potentially be addressed by conditioning the splits on the evolutionary history: if a certain population is split only to be merged again as soon as the maturation period is over, this hints that the population does not need to be split as often. Second, Hyb-CCEA failed to consistently achieve high-performance solutions in the more challenging variant of the soccer task. This was not a problem specific to Hyb-CCEA, as the traditional CCEA achieved significantly inferior results. The results suggest that the coevolutionary process is converging prematurely to mediocre stable states, a problem that we have studied in Chapter 3. The next step of our research is therefore to combine novelty-driven cooperative coevolution with Hyb-CCEA, in an attempt to simultaneously improve scalability and avoid premature convergence. We will discuss this approach in greater detail in Section 7.2.

6.6 Summary

We proposed Hyb-CCEA, an extension of cooperative coevolutionary algorithms that does not rely on a fixed mapping (typically one-to-one) between agents and coevolving populations. In Hyb-CCEA, a population can be dynamically assigned to several agents, therefore enabling partial heterogeneity. By merging populations that are evolving behaviourally similar agents, Hyb-CCEA can find team compositions suitable for solving the given task, along with optimising the agent controllers. We thoroughly explored the properties of Hyb-CCEA in an abstract multiagent task (the coverage task), and then validated the approach with two simulated cooperative multirobot tasks: multi-rover foraging and a soccer task.

We showed that Hyb-CCEA can leverage the advantages of concurrent learning, such as the emergence of specialisations, while at the same time significantly improving the scalability of such algorithms with respect to the number of agents in the team. Hyb-CCEA is a promising new approach for evolving controllers for large teams where there is little knowledge about the task at hand. In Hyb-CCEA, there is no need to specify the team composition (genetically heterogeneous, homogeneous, or any compositions in between), nor the desired specialisations, which represents a significant advance over the current state of the art. The proposed approach optimises the agent controllers for solving the given task, while at the same time converging to a suitable team composition.

Chapter 7

Conclusions

As discussed in Chapter 2, heterogeneous multirobot systems have shown considerable potential in a number of applications (Dorigo et al., 2013; Parker, 2008), presenting several advantages over the more commonly studied homogeneous multirobot systems. Synthesising control for heterogeneous systems is, however, a challenging endeavour, since the search space grows proportionally to the number of different agents (Panait and Luke, 2005a). Cooperative coevolutionary algorithms (CCEAs) (Popovici et al., 2012; Potter and De Jong, 2000) are a promising approach to deal with such large search spaces, but their use is also associated with a number of challenges of their own. Previous works have shown that CCEAs are not necessarily attracted to a global optimum, and often prematurely converge to mediocre stable states (Panait, 2010; Wiegand, 2003); they can be inefficient when applied to a large number of agents (Colby and Tumer, 2015a; D'Ambrosio et al., 2010); and they have only been demonstrated in simulated morphologically homogeneous multirobot systems.

In this thesis, we developed and studied methods for overcoming the aforementioned issues, in order to achieve cooperative coevolutionary algorithms that can be efficiently applied to realistic and complex multirobot tasks. The developed methods are intended to be applicable to any multirobot task, and we strived to obtain general results. To this end, our research was based on empirical studies, conducted mainly in simulation environments, but also on a real robotic system. We did not to focus on solving a specific robotics task, but rather conducted our research based on a number of multirobot domains inspired by previous works from other authors.

7.1 Discussion

In this section, we summarise and discuss the main findings of our work, in light of the initial research questions.

Research Question 1

How can behavioural novelty be leveraged to avoid premature convergence in cooperative coevolutionary algorithms?

In Chapter 3, we proposed novelty-driven cooperative coevolution, the first CCEA based on novelty search, in which individuals are rewarded according to their behavioural novelty, besides the traditional fitness score. We studied three different approaches for defining behavioural novelty: *NS-Team* – the behavioural novelty of an individual corresponds to the novelty of the team with which it was evaluated; *NS-Ind* – the novelty of an individual is the novelty of the agent's individual behaviour, and *NS-Mix* – a combination of the two. In all three approaches,

the novelty score was combined with the traditional fitness score via a Pareto-based multiobjective ranking.

We compared the three approaches with traditional fitness-driven cooperative coevolution, in three simulated multirobot domains: in several variants of a cooperative predator-prey task, ranging from two to seven predators; in a cooperative foraging task with two agents; and in a herding task with four agents. Our results consistently showed that team-level novelty scoring is the most effective approach, and it significantly outperformed fitness-driven coevolution according to multiple performance criteria. *NS-Team* can avoid premature convergence by continuously avoiding stable states – the novelty measure promotes individuals that generate novel team behaviours, countering the pressure of over-fitting to the other coevolving populations.

We compared the novelty-driven approaches with other techniques described in previous works for overcoming premature convergence, namely increasing the number of randomly chosen collaborators that are used for each individual evaluation (Panait, 2010). This strategy revealed to be mostly ineffective, offering little to no advantages in the considered problems, at the cost of significantly increased computational complexity. Novelty-driven cooperative coevolution, on the other hand, could substantially increase the effectiveness of the coevolutionary process, while still requiring a single collaboration to evaluate each individual, which is the common practice in studies that apply CCEAs to multirobot domains. Besides avoiding stable states and improving the quality of the solutions evolved, we also showed that *NS-Team* is capable of discovering a wide diversity of solutions in a single evolutionary run, as opposed to fitness-driven CCEAs that typically converge to a single class of solutions.

In addition to the three domains used in Chapter 3, novelty-driven cooperative coevolution, *NS-Team* in particular, was further validated in the following domains:

- In a preliminary study (Gomes et al., 2014a), we successfully applied *NS-Team* to a simulated keepaway soccer task with three agents.

- In Chapter 4, we applied *NS-Team* to the evolution of controllers for a predator-prey pursuit task, which were then transferred to an aquatic multirobot system, and evaluated in real-world conditions. Besides the success of the teams evolved by *NS-Team*, we showed that the diversity of behaviours that was observed in the simulation environment was also observed in the real robots, thus validating the capability of *NS-Team* to generate diverse and useful solutions.

- In Chapter 5, we used *NS-Team* in a cooperative foraging task with a morphologically heterogeneous systems. We compared it with other techniques that rely on the experimenter's knowledge to overcome premature convergence, namely incremental evolution (Gomez and Miikkulainen, 1997) and multi-objectivisation with sub-goals (Mouret and Doncieux, 2008). The results showed that novelty-driven coevolution was able to match or surpass the performance of the competing approaches, but without relying on the experimenter's biases for task decomposition.

Overall, our results show that promoting individuals that cause novel team behaviours (*NS-Team*) is a successful approach for avoiding premature convergence in CCEAs, which does not significantly increase computational complexity nor does it

rely on the experimenter's prior knowledge, as opposed to other competing techniques. The promising results obtained with a large number and variety of multirobot domains and tasks attest to the general applicability of this approach. Moreover, the widespread use of novelty-based techniques, and successful application outside robotics domains (e.g. Liapis et al., 2015; Martínez et al., 2013; Naredo et al., 2016), suggest that novelty-driven cooperative coevolution might be applicable to the evolution of non-embodied agents, although additional studies would be needed in this direction.

Research Question 2

Can cooperative coevolutionary algorithms be effectively used to evolve controllers for real multirobot systems, and morphologically heterogeneous systems?

The demonstration of CCEAs in the evolution of control for multirobot systems has, so far, been mainly restricted to simulated and morphologically homogeneous systems, leaving out two important types of multirobot systems: real multirobot systems, operating in realistic conditions; and morphologically heterogeneous multirobot systems (Parker, 2008). In this thesis, we studied the challenges on evolving control for these types of systems. We studied standard CCEA algorithms, as well as the proposed novelty-driven cooperative coevolution.

In Chapter 4, we demonstrated the first use of CCEAs to synthesise control for a real multirobot system. Our study was based on a cooperative predator-prey pursuit task with three predators, and the multirobot system was a surface aquatic system (Costa et al., 2016) that had been previously used in other evolutionary robotics studies (Duarte et al., 2016e). The controllers were evolved in simulation, and then systematically evaluated on the real system. To promote transferability, we followed the same approach as previous studies with that system (Duarte et al., 2016b): during evolution, we injected noise in the sensors and actuators; we used multiple simulation trials to evaluate each team, with diverse initial conditions; and we used the same technological platform (including simulator, hardware, and onboard software) that was developed and tuned for previous studies.

Overall, the performance of the teams in simulation matched the performance of the teams when evaluated in the real system. Out of the five different teams tested in the real robots, only one was not able to perform as well as in simulation. This result is highly consistent with the previous results obtained with the same system, with non-coevolutionary techniques (Duarte et al., 2016b). This consistency of results suggests that cooperative coevolutionary algorithms can be successfully applied to real robotic problems without significant specific challenges.

In Chapter 5, we studied the use of CCEAs to synthesise control for a highly morphologically heterogeneous system. Our study relied on the *aerial-ground foraging task*, where a simple ground robot had to cooperate with a much more complex and capable aerial robot to find a collect items in the arena. We used different task variants where we varied the complexity of skills the robots had to learn before they could effectively cooperate with one another. We showed that when the robots can easily establish cooperation with one another, right from the beginning of the evolutionary process, the CCEA can achieve successful teams despite the stark differences in morphology and controller complexity between the agents. In the task variants where cooperation was harder to achieve, CCEAs often converged to low performing non-cooperative solutions, which correspond to stable states from which it is difficult to escape. Our results highlight the importance of sustaining a mutual development of skills in the coevolving agents (Uchibe et al., 1998), and the importance

of facilitating cooperation when it is fundamental for the solution of the task. When such conditions are met, the coevolutionary process is able to afford an arbitrary degree of heterogeneity in the coevolving populations.

These two studies with a real multirobot system and a morphologically heterogeneous system, join the large number of previous studies based on simulated morphologically homogeneous systems. In general, the existing demonstrations support that CCEAs are applicable to a wide range of multirobot tasks.

Research Question 3

Can the scalability of cooperative coevolutionary algorithms be improved through dynamic team heterogeneity?

In Chapter 6, we proposed Hyb-CCEA, a CCEA extension aimed at improving the scalability of the coevolutionary process with respect to the number of agents in the team. Hyb-CCEA puts the team composition under evolutionary control, departing from the traditional one-to-one mapping between agents and populations. In Hyb-CCEA, a single population can be dynamically assigned to several agents, therefore enabling partial heterogeneity. During evolution, populations that are evolving behaviourally similar agents can be merged, decreasing the heterogeneity of the system. Complementary, populations can be stochastically split for increased heterogeneity, thus ensuring the exploration of different team compositions throughout the evolutionary process.

We empirically evaluated Hyb-CCEA in a total of four domains: an abstract function optimisation domain, which allowed us to use a variety of problem instances for studying the core properties of Hyb-CCEA; two multi-rover foraging tasks with ten robots; two soccer tasks with teams of five robots; and six herding tasks with teams composed of five to ten robots (Gomes et al., 2015b). Our results showed that Hyb-CCEA can find suitable team compositions for solving the given task, ranging from fully heterogeneous to fully homogeneous. As such, it can significantly decrease the number of agent controllers that need to be evolved, thereby reducing the amount of redundant learning (separate populations evolving the same behaviours). These advantages resulted in a large reduction in the number of evaluations needed to achieve solutions. Moreover, Hyb-CCEA was often able to achieve solutions of a quality never reached by a traditional fully heterogeneous CCEA, which might be explained by the mitigation of credit assignment issues — if each population is assigned to multiple agents, the modification of a single individual can have a greater impact in the team's performance, which in turn leads to more clear fitness gradients.

While previous works with non-coevolutionary algorithms had shown the potential of dynamic team heterogeneity to improve multiagent learning (Bongard, 2000; D'Ambrosio and Stanley, 2008; Hara, 1999), to the best of our knowledge, Hyb-CCEA is the first coevolutionary algorithm to enable that. We have shown that departing from the fixed one-to-one mapping between agents and populations can be a beneficial approach, as it allows scalability issues of CCEAs to be addressed, such as redundant learning and credit assignment issues. At the same time, the key advantages of CCEAs are preserved, namely the ability to work on a decomposition of the problem and the emergence of agent specialisations.

Final Remarks

First and foremost, this thesis confirms the potential of cooperative coevolutionary algorithms as a powerful tool to evolve control for heterogeneous multirobot

systems. We shed a new light on how CCEAs can be improved, mitigating fundamental issues specific to cooperative coevolutionary algorithms, namely premature convergence and poor scalability with respect to the number of agents. The methods proposed in this thesis follow the recent research trend concerned with moving evolutionary robotics techniques beyond pure black-box optimisation (Doncieux and Mouret, 2014; Silva et al., 2016b). By leveraging insights on the behaviours that are being produced by the evolutionary algorithm, we were able to avoid premature convergence to mediocre stable states (novelty-driven cooperative coevolution), and reduce the amount of redundant learning in the coevolutionary process (Hyb-CCEA).

This thesis advances the state of the art in the direction of methods for efficiently synthesising control for heterogeneous multirobot systems, capable of solving real-world tasks. The ability to produce control for such systems, which is currently a challenge for both automatic and manual techniques, is a fundamental step to embrace heterogeneity in multirobot systems, and realise the full potential of these systems.

7.2 Future Work

In this section, we discuss the main limitations of this work, and promising future directions.

Novelty-driven CCEA with Generic Behaviour Characterisations The experiments with novelty-driven cooperative coevolution reported in this thesis relied on task-specific behaviour characterisations, provided by the experimenter. We adopted this approach since it is arguably the simplest to implement and analyse, and it is by far the most commonly used in novelty search studies (Gomes et al., 2014e). Although the definition of behavioural measures did not pose a problem in our tasks, in future work, we will experiment with task-independent (generic) behaviour characterisations (Doncieux and Mouret, 2010; Gomes and Christensen, 2013; Meyerson et al., 2016), and other techniques that make novelty less sensitive to the choice of the behaviour characterisation (Doncieux and Mouret, 2013).

Novelty-driven Hyb-CCEA In the experiments with Hyb-CCEA, we identified a problem of premature convergence in one of the tasks. In Chapter 3, we showed that premature convergence problems in CCEAs can be mitigated with novelty-driven coevolution. In future work, we will empirically evaluate the combination of novelty-driven coevolution and the dynamic team heterogeneity provided by Hyb-CCEA. These two techniques target different parts of the evolutionary algorithm and are algorithmically compatible with one another. There are, however, two potential issues that still need to be studied:

- It is unclear if and how the evolutionary dynamics of novelty search will influence Hyb-CCEA, and vice-versa. For instance, the behavioural novelty pressure could push towards more heterogeneous teams, as in the short term the behavioural possibilities increase with the heterogeneity of the team, possibly preventing the formation of homogeneous sub-teams. On the other hand, having homogeneous sub-teams can facilitate the achievement of novel team behaviours, as a modification in a single controller can have a greater impact in the teams' behaviour.

- Having to specify both a team-level behaviour characterisation (for novelty-driven coevolution) and an agent-level characterisation (for Hyb-CCEA) can be burdensome. We will therefore study how to automatically derive team-level characterisations from agent-level characterisations, or vice-versa. Our previous work with generic characterisations (Gomes and Christensen, 2013) suggests that this is indeed possible.

Improvement of Hyb-CCEA Operators The proposed implementation of Hyb-CCEA demonstrates the potential of the approach using relatively simple operators. We contend, however, that the algorithm could be improved in further studies. We identified three main topics where such work could focus:

- Due to the periodic and purely stochastic population splits, the number of populations in the algorithm is on average above the number of populations needed to optimally solve the task. This could potentially be addressed by conditioning the splits on the evolutionary history: if a certain population is split only to be merged again as soon as the maturation period is over, this hints that the population may not need to be split as often.

- The current implementation does not support morphologically heterogeneous systems, as it assumes that all agents can exchange their controllers. A straightforward way of overcoming this issue would be to restrict population merges and splits to morphologically similar agents.

- Since Hyb-CCEA enables the formation of homogeneous sub-teams, it would be valuable to study whether such teams could be scaled with respect to the number of agents after the evolutionary process is finished. Such team size adaptation has been shown valuable in other studies (D'Ambrosio et al., 2010).

Emergent Morphological Specialisation Morphological heterogeneity in multi-robot systems can offer considerable advantages (Dorigo et al., 2013; Parker, 2008), as our experiments in Chapter 5 also showed. In previous works, morphological heterogeneity is defined by the experimenter, typically corresponding to the types of robots available. A promising line of work is to study if and how CCEAs can be used to evolve morphological specialisation along with behavioural specialisation. Certain morphological features of the robots could be placed under evolutionary control, such as the type, number and placement of the sensors. The coevolutionary process would then be able to assign different capabilities to different robots, ideally evolving a complementary set of capabilities. Besides potentially providing an advantage in domains where robots with significantly different capabilities are required, it has also been shown that the gradual increase in morphological complexity in a single-robot system can actually facilitate the evolution of robust behaviours (Bongard, 2011).

Bibliography

Agogino, A. and Tumer, K. (2004). "Efficient evaluation functions for multi-rover systems". In: *Genetic and Evolutionary Computation Conference (GECCO)*. Springer, pp. 1–11.

Agogino, A. and Tumer, K. (2007). "Evolving distributed agents for managing air traffic". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 1888–1895.

Agogino, A. and Tumer, K. (2008). "Efficient evaluation functions for evolving coordination". In: *Evolutionary Computation* 16 (2), pp. 257–288.

Anderson, C. and Franks, N. R. (2001). "Teams in animal societies". In: *Behavioral Ecology* 12 (5), pp. 534–540.

Arai, T., Pagello, E., and Parker, L. E. (2002). "Editorial: Advances in multi-robot systems". In: *IEEE Transactions on Robotics and Automation* 18 (5), pp. 655–661.

Balch, T. (1998). "Behavioral diversity in learning robot teams". PhD thesis. Georgia Institute of Technology.

Balch, T. and Arkin, R. C. (1998). "Behavior-based formation control for multirobot teams". In: *IEEE Transactions on Robotics and Automation* 14 (6), pp. 926–939.

Barrett, S. and Stone, P. (2015). "Cooperating with Unknown Teammates in Complex Domains: A Robot Soccer Case Study of Ad Hoc Teamwork". In: *AAAI Conference on Artificial Intelligence*. AAAI Press, pp. 2010–2016.

Bayındır, L. (2016). "A review of swarm robotics tasks". In: *Neurocomputing* 172, pp. 292–321.

Beer, R. D. and Gallagher, J. C. (1992). "Evolving Dynamical Neural Networks for Adaptive Behavior". In: *Adaptive Behavior* 1 (1), pp. 91–122.

Bernard, A., André, J.-B., and Bredeche, N. (2015). "Evolution of cooperation in evolutionary robotics: the tradeoff between evolvability and efficiency". In: *European Conference on Artificial Life (ECAL 2015)*, pp. 495–502.

Blumenthal, H. J. and Parker, G. B. (2004). "Co-evolving team capture strategies for dissimilar robots". In: *AAAI Artificial Multiagent Learning Symposium*. Vol. 2. AAAI Press.

Bongard, J. (2011). "Morphological change in machines accelerates the evolution of robust behavior". In: *Proceedings of the National Academy of Sciences* 108 (4), pp. 1234–1239.

Bongard, J. C. (2000). "The legion system: A novel approach to evolving heterogeneity for collective problem solving". In: *Genetic Programming*. Vol. 1802. LNCS. Springer, pp. 16–28.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). "Swarm robotics: a review from the swarm engineering perspective". In: *Swarm Intelligence* 7 (1), pp. 1–41.

Bucci, A. and Pollack, J. B. (2002). "A Mathematical Framework for the Study of Coevolution." In: *Foundations of Genetic Algorithms (FOGA)*. Vol. 7. Morgan Kaufmann, pp. 221–235.

Campbell, A. and Wu, A. S. (2011). "Multi-agent role allocation: issues, approaches, and multiple perspectives". In: *Autonomous Agents & Multi-Agent Systems* 22 (2), pp. 317–355.

Candea, C., Hu, H., Iocchi, L., Nardi, D., and Piaggio, M. (2001). "Coordination in multi-agent RoboCup teams". In: *Robotics and Autonomous Systems* 36 (2), pp. 67–86.

Cao, Y. U., Fukunaga, A. S., and Kahng, A. (1997). "Cooperative mobile robotics: Antecedents and directions". In: *Autonomous robots* 4 (1), pp. 7–27.

Castelli, M., Manzoni, L., and Vanneschi, L. (2011). "A Method to Reuse Old Populations in Genetic Algorithms". In: *Portuguese Conference on Artificial Intelligence (EPIA)*. Vol. 7026. LNCS. Springer, pp. 138–152.

Chaimowicz, L., Cowley, A., Gomez-Ibanez, D., Grocholsky, B., Hsieh, M. A., Hsu, H., Keller, J. F., Kumar, V., Swaminathan, R., and Taylor, C. J. (2005). "Deploying Air-Ground Multi-Robot Teams in Urban Environments". In: *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III: Proceedings from the 2005 International Workshop on Multi-Robot Systems*. Ed. by L. E. Parker, F. E. Schneider, and A. C. Schultz. Springer, pp. 223–234.

Christensen, A. L. and Dorigo, M. (2006). "Incremental Evolution of Robot Controllers for a Highly Integrated Task". In: *International Conference on Simulation of Adaptive Behavior (SAB)*. Vol. 4095. LNCS. Springer, pp. 473–484.

Christensen, A. L., Duarte, M., Costa, V., Rodrigues, T., Gomes, J., Silva, F., and Oliveira, S. M. (2016). "A Sea of Robots". In: *AAAI Conference on Artificial Intelligence*. AAAI Press. URL: `https://www.youtube.com/watch?v=JBrkszUnms8`.

Colby, M. and Tumer, K. (2015a). "An evolutionary game theoretic analysis of difference evaluation functions". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 1391–1398.

Colby, M. and Tumer, K. (2015b). "Fitness function shaping in multiagent cooperative coevolutionary algorithms". In: *Autonomous Agents & Multi-Agent Systems*, pp. 1–28.

Costa, V., Duarte, M., Rodrigues, T., Oliveira, S. M., and Christensen, A. L. (2016). "Design and Development of an Inexpensive Aquatic Swarm Robotics System". In: *IEEE/MTS OCEANS*. IEEE Press, pp. 1–7.

Şahin, E. (2005). "Swarm Robotics: From Sources of Inspiration to Domains of Application". In: *Swarm Robotics*. Vol. 3342. LNCS. Springer, pp. 10–20.

Cuccu, G. and Gomez, F. J. (2011). "When Novelty Is Not Enough". In: *European Conference on the Applications of Evolutionary Computation (EvoApps)*. Vol. 6624. LNCS. Springer, pp. 234–243.

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. (2015). "Robots that can Adapt like Animals". In: *Nature* 521 (7553), pp. 503–507.

D'Ambrosio, D. B. and Stanley, K. O. (2008). "Generative encoding for multiagent learning". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 819–826.

D'Ambrosio, D. B., Lehman, J., Risi, S., and Stanley, K. O. (2010). "Evolving policy geometry for scalable multiagent learning". In: *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*. IFAAMAS, pp. 731–738.

Daniel, T., Manley, J., and Trenaman, N. (2011). "The Wave Glider: enabling a new approach to persistent ocean observation and research". In: *Ocean Dynamics* 61 (10), pp. 1509–1520.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6 (2), pp. 182–197.

Didi, S. and Nitschke, G. (2016). "Multi-agent Behavior-Based Policy Transfer". In: *European Conference on the Applications of Evolutionary Computation (EvoApps)*. Springer, pp. 181–197.

Doncieux, S. and Mouret, J.-B. (2013). "Behavioral diversity with multiple behavioral distances". In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press, pp. 1427–1434.

Doncieux, S. and Mouret, J.-B. (2010). "Behavioral diversity measures for Evolutionary Robotics". In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press, pp. 1–8.

Doncieux, S. and Mouret, J.-B. (2014). "Beyond black-box optimization: a review of selective pressures for evolutionary robotics". In: *Evolutionary Intelligence* 7 (2), pp. 71–93.

Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. G. (2015). "Evolutionary Robotics: What, Why, and Where to". In: *Frontiers in Robotics and AI* 2, p. 4.

Dorigo, M., Floreano, D., Gambardella, L., Mondada, F., et al. (2013). "Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms". In: *IEEE Robotics & Automation Magazine* 20 (4), pp. 60–71.

Duan, H. B. and Liu, S. Q. (2010). "Unmanned air/ground vehicles heterogeneous cooperative techniques: Current status and prospects". In: *Science China Technological Sciences* 53 (5), pp. 1349–1355.

Duarte, M., Silva, F., Rodrigues, T., Oliveira, S. M., and Christensen, A. L. (2014). "JBotEvolver: A Versatile Simulation Platform for Evolutionary Robotics". In: *International Conference on the Synthesis and Simulation of Living Systems (ALife)*. MIT Press, pp. 210–211.

Duarte, M., Gomes, J., Costa, V., Rodrigues, T., Silva, F., Lobo, V., Marques, M., Oliveira, S. M., and Christensen, A. L. (2016a). "Application of Swarm Robotic Systems to Marine Environmental Monitoring". In: *IEEE/MTS OCEANS*. IEEE Press, pp. 1–8.

Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., and Christensen, A. L. (2016b). "Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots". In: *PLoS ONE* 11 (3), e0151834.

Duarte, M., Gomes, J., Oliveira, S. M., and Christensen, A. L. (2016c). "EvoRBC: Evolutionary Repertoire-based Control for Robots with Arbitrary Locomotion Complexity". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 93–100.

Duarte, M., Gomes, J., Costa, V., Oliveira, S. M., and Christensen, A. L. (2016d). "Hybrid Control for a Real Swarm Robotics System in an Intruder Detection Task". In: *European Conference on the Applications of Evolutionary Computation (EvoApps)*. Springer, pp. 213–230.

Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., and Christensen, A. L. (2016e). "Unleashing the Potential of Evolutionary Swarm Robotics in the Real World". In: *Genetic and Evolutionary Computation Conference Companion (GECCO)*. ACM Press, pp. 159–160.

Duarte, M., Gomes, J., Oliveira, S. M., and Christensen, A. L. (2017). "Evolution of Repertoire-based Control for Robots with Complex Locomotor Systems". In: *IEEE Transactions on Evolutionary Computation*. Under revision.

Ducatelle, F., Di Caro, G., Pinciroli, C., and Gambardella, L. (2011). "Self-organized cooperation between robotic swarms". In: *Swarm Intelligence* 5 (2), pp. 73–96.

Eiben, A. E. (2014). "Grand Challenges for Evolutionary Robotics". In: *Frontiers in Robotics and AI* 1, p. 4.

Elman, J. L. (1990). "Finding Structure in Time". In: *Cognitive Science* 14 (2), pp. 179–211.

Fehérvári, I. and Elmenreich, W. (2010). "Evolving neural network controllers for a team of self-organizing robots". In: *Journal of Robotics* 2010, p. 841286.

Ferrante, E., Turgut, A. E., Duéñez-Guzmán, E., Dorigo, M., and Wenseleers, T. (2015). "Evolution of Self-Organized Task Specialization in Robot Swarms". In: *PLoS Computational Biology* 11 (8), e1004273.

Floreano, D. and Mondada, F. (1994). "Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot". In: *Simulation of Adaptive Behavior (SAB)*. MIT Press, 421–430.

Floreano, D., Dürr, P., and Mattiussi, C. (2008). "Neuroevolution: from architectures to learning". In: *Evolutionary Intelligence* 1 (1), pp. 47–62.

Francesca, G. and Birattari, M. (2016). "Automatic Design of Robot Swarms: Achievements and Challenges". In: *Frontiers in Robotics and AI* 3, p. 29.

Goldberg, D. E. and Richardson, J. (1987). "Genetic algorithms with sharing for multimodal function optimization". In: *Genetic Algorithms and their Applications: Second International Conference on Genetic Algorithms*. Lawrence Erlbaum, pp. 41–49.

Gomes, J. (2014). "Evolution of heterogeneous multirobot systems through behavioural diversity". In: *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*. IFAAMAS, pp. 1729–1730.

Gomes, J. and Christensen, A. L. (2013). "Generic Behaviour Similarity Measures for Evolutionary Swarm Robotics". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 199–206.

Gomes, J., Urbano, P., and Christensen, A. L. (2012). "Progressive Minimal Criteria Novelty Search". In: *Ibero-American Conference on Artificial Intelligence (IBERAMIA)*. Vol. 7637. LNAI. Springer, pp. 281–290.

Gomes, J., Urbano, P., and Christensen, A. L. (2013). "Evolution of swarm robotics systems with novelty search". In: *Swarm Intelligence* 7 (2–3), pp. 115–144.

Gomes, J., Mariano, P., and Christensen, A. L. (2014a). "Avoiding Convergence in Cooperative Coevolution with Novelty Search". In: *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*. IFAAMAS, pp. 1149–1156.

Gomes, J., Mariano, P., and Christensen, A. L. (2014b). "Diversity-based Coevolution of Behaviourally Heterogeneous Multirobot Systems". In: *Workshop on Nature-inspired Techniques for Robotics at PPSN*.

Gomes, J., Mariano, P., and Christensen, A. (2014c). "Novelty Search in Competitive Coevolution". In: *Parallel Problem Solving from Nature (PPSN)*. Vol. 8672. LNCS. Springer, pp. 233–242.

Gomes, J., Urbano, P., and Christensen, A. L. (2014d). "PMCNS: Using a Progressively Stricter Fitness Criterion to Guide Novelty Search". In: *International Journal of Natural Computing Research* 4, pp. 1–19.

Gomes, J., Mariano, P., and Christensen, A. L. (2014e). "Systematic Derivation of Behaviour Characterisations in Evolutionary Robotics". In: *International Conference on the Synthesis and Simulation of Living Systems (ALife)*. MIT Press, pp. 212–219.

Gomes, J., Mariano, P., and Christensen, A. L. (2015a). "Cooperative Coevolution of Morphologically Heterogeneous Robots". In: *European Conference on Artificial Life (ECAL)*. MIT Press, pp. 312–319.

Gomes, J., Mariano, P., and Christensen, A. L. (2015b). "Cooperative Coevolution of Partially Heterogeneous Multiagent Systems". In: *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*. IFAAMAS, pp. 297–305.

Gomes, J., Mariano, P., and Christensen, A. L. (2015c). "Devising Effective Novelty Search Algorithms: A Comprehensive Empirical Study". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 943–950.

Gomes, J., Mariano, P., and Christensen, A. L. (2015d). "Hyb-CCEA: Cooperative Coevolution of Hybrid Teams". In: *Genetic and Evolutionary Computation Conference Companion (Evolving Collective Behaviors in Robotics Workshop)*. ACM Press, pp. 1251–1252.

Gomes, J., Mariano, P., and Christensen, A. L. (2016a). "Challenges in cooperative coevolution of physically heterogeneous robot teams". In: *Natural Computing*, pp. 1–18.

Gomes, J., Duarte, M., Mariano, P., and Christensen, A. L. (2016b). "Cooperative Coevolution of Control for a Real Multirobot System". In: *Parallel Problem Solving from Nature (PPSN)*. Springer, pp. 591–601.

Gomes, J., Mariano, P., and Christensen, A. L. (2017a). "Dynamic Team Heterogeneity in Cooperative Coevolutionary Algorithms". In: *IEEE Transactions on Evolutionary Computation*. Under revision.

Gomes, J., Mariano, P., and Christensen, A. L. (2017b). "Novelty-driven Cooperative Coevolution". In: *Evolutionary Computation*. *In press*.

Gomez, F. and Miikkulainen, R. (1997). "Incremental Evolution of Complex General Behavior". In: *Adaptive Behavior* 5 (3–4), pp. 317–342.

Gomez, F. J. (2009). "Sustaining diversity using behavioral information distance". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 113–120.

Grabowski, R., Navarro-Serment, L. E., Paredis, C. J., and Khosla, P. K. (2000). "Heterogeneous teams of modular robots for mapping and exploration". In: *Autonomous Robots* 8 (3), pp. 293–308.

Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2006). "Autonomous Self-Assembly in Swarm-Bots". In: *IEEE Transactions on Robotics* 22 (6), pp. 1115–1130.

Guo, Y., Parker, L. E., and Madhavan, R. (2004). "Towards collaborative robots for infrastructure security applications". In: *International Symposium on Collaborative Technologies and Systems (CTS)*. Vol. 2004. Society for Modeling and Simulation International, pp. 235–240.

Hara, A. (1999). "Emergence of cooperative behavior using ADG; Automatically Defined Groups". In: *Genetic and Evolutionary Computation Conference (GECCO)*. Morgan Kaufmann, pp. 1039–1046.

Harvey, I., Husbands, P., and Cliff, D. (1993). "Issues in evolutionary robotics". In: *International Conference on Simulation of Adaptive Behavior (SAB)*. MIT Press, pp. 364–373.

Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N. (1997). "Evolutionary robotics: the Sussex approach". In: *Robotics and autonomous systems* 20 (2-4), pp. 205–224.

Haynes, T. and Sen, S. (1997). "Crossover operators for evolving a team". In: *Genetic programming* 199.

Hornby, G. (2006). "ALPS: the age-layered population structure for reducing the problem of premature convergence". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 815–822.

Howard, A., Parker, L. E., and Sukhatme, G. S. (2006). "Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection". In: *International Journal of Robotics Research* 25 (5-6), pp. 431–447.

Hsieh, M. A., Cowley, A., Keller, J. F., Chaimowicz, L., Grocholsky, B., Kumar, V., Taylor, C. J., Endo, Y., Arkin, R. C., Jung, B., et al. (2007). "Adaptive teams of

autonomous aerial and ground robots for situational awareness". In: *Journal of Field Robotics* 24 (11-12), pp. 991–1014.

Hu, J., Goodman, E. D., Seo, K., Fan, Z., and Rosenberg, R. (2005). "The Hierarchical Fair Competition (HFC) Framework for Sustainable Evolutionary Algorithms". In: *Evolutionary Computation* 13 (2), pp. 241–277.

Hutter, M. and Legg, S. (2006). "Fitness uniform optimization". In: *IEEE Transactions on Evolutionary Computation* 10 (5), pp. 568–589.

Hwang, C.-L. and Masud, A. S. M. (2012). *Multiple objective decision making—methods and applications: a state-of-the-art survey*. Vol. 164. Springer.

Iba, H. (1996). "Emergent cooperation for multiple agents using genetic programming". In: *Parallel Problem Solving from Nature (PPSN)*. Vol. 1141. LNCS. Springer, pp. 32–41.

Inden, B., Jin, Y., Haschke, R., Ritter, H., and Sendhoff, B. (2013). "An examination of different fitness and novelty based selection methods for the evolution of neural networks". In: *Soft Computing* 17 (5), pp. 753–767.

Iocchi, L., Nardi, D., Piaggio, M., and Sgorbissa, A. (2003). "Distributed coordination in heterogeneous multi-robot systems". In: *Autonomous Robots* 15 (2), pp. 155–168.

Jakobi, N. (1997). "Evolutionary robotics and the radical envelope-of-noise hypothesis". In: *Adaptive Behavior* 6 (2), pp. 325–368.

Jansen, T. and Wiegand, R. P. (2003). "Exploring the explorative advantage of the cooperative coevolutionary (1+1) EA". In: *Genetic and Evolutionary Computation Conference (GECCO)*. Vol. 2723. LNCS. Springer, pp. 310–321.

Jansen, T. and Wiegand, R. P. (2004). "The cooperative coevolutionary (1+1) EA". In: *Evolutionary Computation* 12 (4), pp. 405–434.

Jones, C. V. and Mataric, M. J. (2005). "Behavior-Based Coordination in Multi-Robot Systems". In: *Autonomous Mobile Robots: Sensing, Control, Decision Making, and Applications*. Marcel Dekker, 549–569.

Jones, E. G., Browning, B., Dias, M. B., Argall, B., Veloso, M., and Stentz, A. (2006). "Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, pp. 570–575.

Jones, J. L. (2006). "Robots at the tipping point: the road to iRobot Roomba". In: *IEEE Robotics & Automation Magazine* 13 (1), pp. 76–78.

Jones, T. and Forrest, S. (1995). "Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms". In: *International Conference on Genetic Algorithms (ICGA)*. Morgan Kaufmann, pp. 184–192.

Jordan, M. I. (1997). "Serial order: A parallel distributed processing approach". In: *Neural-Network Models of Cognition Biobehavioral Foundations*. Vol. 121. Advances in Psychology. North-Holland, pp. 471–495.

Kengyel, D., Hamann, H., Zahadat, P., Radspieler, G., Wotawa, F., and Schmickl, T. (2015). "Potential of heterogeneity in collective behaviors: A case study on heterogeneous swarms". In: *Principles and Practice of Multi-Agent Systems*. Springer, pp. 201–217.

Kistemaker, S. and Whiteson, S. (2011). "Critical factors in the performance of novelty search". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 965–972.

Knowles, J., Watson, R., and Corne, D. (2001). "Reducing Local Optima in Single-Objective Problems by Multi-objectivization". In: *Evolutionary Multi-Criterion Optimization*. Vol. 1993. LNCS. Springer, pp. 269–283.

Knudson, M. and Tumer, K. (2010). "Coevolution of heterogeneous multi-robot teams". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 127–134.

Kohonen, T. (1990). "The self-organizing map". In: *Proceedings of the IEEE* 78 (9), pp. 1464–1480.

Koos, S., Mouret, J.-B., and Doncieux, S. (2013). "The transferability approach: Crossing the reality gap in evolutionary robotics". In: *IEEE Transactions on Evolutionary Computation* 17 (1), pp. 122–145.

Kose, H., Kaplan, K., Mericli, C., Tatlidede, U., and Akin, L. (2005). "Market-driven multi-agent collaboration in robot soccer domain". In: *Cutting Edge Robotics*. In-Tech, pp. 407–416.

Lacroix, S. and Le Besnerais, G. (2011). "Issues in Cooperative Air/Ground Robotic Systems". In: *Robotics Research*. Vol. 66. Springer Tracts in Advanced Robotics. Springer, pp. 421–432.

Lehman, J. and Stanley, K. O. (2010). "Revising the evolutionary computation abstraction: minimal criteria novelty search". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 103–110.

Lehman, J. and Stanley, K. O. (2011a). "Abandoning Objectives: Evolution Through the Search for Novelty Alone". In: *Evolutionary Computation* 19 (2), pp. 189–223.

Lehman, J. and Stanley, K. O. (2011b). "Evolving a diversity of virtual creatures through novelty search and local competition". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 211–218.

Lehman, J., Stanley, K. O., and Miikkulainen, R. (2013). "Effective diversity maintenance in deceptive domains". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 215–222.

Levi, P. and Kernbach, S. (2010). "Heterogeneous Multi-Robot Systems". In: *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*. Springer, pp. 79–163.

Li, L., Martinoli, A., and Abu-Mostafa, Y. S. (2002). "Emergent Specialization in Swarm Systems". In: *Intelligent Data Engineering and Automated Learning (IDEAL)*. Springer, pp. 261–266.

Liapis, A., Yannakakis, G. N., and Togelius, J. (2015). "Constrained novelty search: A study on game content generation". In: *Evolutionary computation* 23 (1), pp. 101–129.

Lichocki, P., Wischmann, S., Keller, L., and Floreano, D. (2013). "Evolving team compositions by agent swapping". In: *IEEE Transactions on Evolutionary Computation* 17 (2), pp. 282–298.

Luke, S. and Spector, L. (1996). "Evolving teamwork and coordination with genetic programming". In: *Genetic Programming*. MIT Press, pp. 150–156.

Luke, S., Hohn, C., Farris, J., Jackson, G., and Hendler, J. (1998). "Co-evolving soccer softbot team coordination with genetic programming". In: *RoboCup-97: Robot Soccer World Cup*. Vol. 1395. LNCS. Springer, pp. 398–411.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). "Mason: A multiagent simulation environment". In: *Simulation* 81 (7), pp. 517–527.

Martínez, Y., Naredo, E., Trujillo, L., and Galván-López, E. (2013). "Searching for novel regression functions". In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press, pp. 16–23.

Matarić, M. and Cliff, D. (1996). "Challenges in evolving controllers for physical robots". In: *Robotics and autonomous systems* 19 (1), pp. 67–83.

Mathews, N., Christensen, A. L., O'Grady, R., and Dorigo, M. (2010). "Cooperation in a heterogeneous robot swarm through spatially targeted communication". In: *Swarm Intelligence*. Vol. 6234. LNCS. Springer, pp. 400–407.

Meyerson, E., Lehman, J., and Miikkulainen, R. (2016). "Learning Behavior Characterizations for Novelty Search". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 149–156.

Miconi, T. (2003). "When evolving populations is better than coevolving individuals: The blind mice problem". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, pp. 647–652.

Miglino, O., Lund, H. H., and Nolfi, S. (1995). "Evolving mobile robots in simulated and real environments". In: *Artificial life* 2 (4), pp. 417–434.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). "The e-puck, a robot designed for education in engineering". In: *International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. Vol. 1. IPCB: Instituto Politécnico de Castelo Branco, pp. 59–65.

Montanier, J.-M., Carrignon, S., and Bredeche, N. (2016). "Behavioral Specialization in Embodied Evolutionary Robotics: Why So Difficult?" In: *Frontiers in Robotics and AI* 3, p. 38.

Mouret, J.-B. (2011). "Novelty-based multiobjectivization". In: *New Horizons in Evolutionary Robotics*. Vol. 341. Studies in Computation Intelligence. Springer, pp. 139–154.

Mouret, J. and Clune, J. (2015). "Illuminating search spaces by mapping elites". In: *CoRR* abs/1504.04909. URL: http://arxiv.org/abs/1504.04909.

Mouret, J.-B. and Doncieux, S. (2008). "Incremental evolution of animats' behaviors as a multi-objective optimization". In: *International Conference on Simulation of Adaptive Behavior (SAB)*. Vol. 5040. LNCS. Springer, pp. 210–219.

Mouret, J.-B. and Doncieux, S. (2012). "Encouraging Behavioral Diversity in Evolutionary Robotics: An Empirical Study". In: *Evolutionary Computation* 20 (1), pp. 91–133.

Murciano, A., Millán, J. d. R., and Zamora, J. (1997). "Specialization in multi-agent systems through learning". In: *Biological Cybernetics* 76 (5), pp. 375–382.

Naredo, E. and Trujillo, L. (2013). "Searching for novel clustering programs". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 1093–1100.

Naredo, E., Trujillo, L., and Martínez, Y. (2013). "Searching for novel classifiers". In: *Genetic Programming (EuroGP)*. Vol. 7831. LNCS. Springer, pp. 145–156.

Naredo, E., Duarte Villaseñor, M. A., García Ortega, M. d. J., Vázquez López, C. E., Trujillo, L., and Siordia, O. S. (2016). "Novelty Search for the Synthesis of Current Followers". In: *Computación y Sistemas* 20 (4).

Nelson, A. L., Grant, E., and Henderson, T. C. (2004). "Evolution of neural controllers for competitive game playing with teams of mobile robots". In: *Robotics and Autonomous Systems* 46 (3), pp. 135–150.

Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). "Fitness functions in evolutionary robotics: A survey and analysis". In: *Robotics and Autonomous Systems* 57 (4), pp. 345–370.

Nitschke, G. (2005a). "Designing emergent cooperation: a pursuit–evasion game case study". In: *Artificial Life and Robotics* 9 (4), pp. 222–233.

Nitschke, G. (2005b). "Emergence of cooperation: State of the art". In: *Artificial Life* 11 (3), pp. 367–396.

Nitschke, G. (2012). "Behavioral heterogeneity, cooperation, and collective construction". In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press, pp. 1–8.

Nitschke, G. S. (2008). "Neuro-evolution for emergent specialization in collective behavior systems". PhD thesis. Vrije Universiteit Amsterdam.

Nitschke, G. S., Schut, M. C., and Eiben, A. E. (2010). "Collective neuro-evolution for evolving specialized sensor resolutions in a multi-rover task". In: *Evolutionary Intelligence* 3 (1), pp. 13–29.

Nitschke, G. S., Schut, M. C., and Eiben, A. E. (2012a). "Evolving behavioral specialization in robot teams to solve a collective construction task". In: *Swarm and Evolutionary Computation* 2, pp. 25–38.

Nitschke, G. S., Eiben, A. E., and Schut, M. C. (2012b). "Evolving team behaviors with specialization". In: *Genetic Programming and Evolvable Machines* 13 (4), pp. 493–536.

Nolfi, S. (2012). "Co-evolving predator and prey robots". In: *Adaptive Behavior* 20 (1), pp. 10–15.

Nolfi, S. and Floreano, D. (2000). *Evolutionary robotics*. MIT Press.

Panait, L. (2010). "Theoretical Convergence Guarantees for Cooperative Coevolutionary Algorithms". In: *Evolutionary Computation* 18 (4), pp. 581–615.

Panait, L. and Luke, S. (2005a). "Cooperative Multi-Agent Learning: The State of the Art". In: *Autonomous Agents & Multi-Agent Systems* 11 (3), pp. 387–434.

Panait, L. and Luke, S. (2005b). "Time-dependent collaboration schemes for cooperative coevolutionary algorithms". In: *AAAI Fall Symposium on Coevolutionary and Coadaptive Systems*. AAAI Press.

Panait, L., Wiegand, R. P., and Luke, S. (2003). "Improving Coevolutionary Search for Optimal Multiagent Behaviors". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, pp. 653–660.

Panait, L., Wiegand, R. P., and Luke, S. (2004). "A Visual Demonstration of Convergence Properties of Cooperative Coevolution". In: *Parallel Problem Solving from Nature (PPSN)*. Vol. 3242. LNCS. Springer, pp. 892–901.

Panait, L., Luke, S., and Harrison, J. F. (2006a). "Archive-based cooperative coevolutionary algorithms". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 345–352.

Panait, L., Luke, S., and Wiegand, R. P. (2006b). "Biasing Coevolutionary Search for Optimal Multiagent Behaviors". In: *IEEE Transactions on Evolutionary Computation* 10 (6), pp. 629–645.

Parker, G. B. and Blumenthal, J. (2002). "Sampling the Nature of A Population: Punctuated Anytime Learning For Co-Evolving A Team". In: *Intelligent Engineering Systems Through Artificial Neural Networks*. ASME, pp. 207–212.

Parker, L., Kannan, B., Tang, F., and Bailey, M. (2004). "Tightly-coupled navigation assistance in heterogeneous multi-robot teams". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, pp. 1016–1022.

Parker, L. E. (1994). *Heterogeneous multi-robot cooperation*. Tech. rep. AITR-1465. MIT Artificial Intelligence Laboratory.

Parker, L. E. (1998). "ALLIANCE: An architecture for fault tolerant multirobot cooperation". In: *IEEE Transactions on Robotics and Automation* 14 (2), pp. 220–240.

Parker, L. E. (2008). "Multiple Mobile Robot Systems". In: *Springer Handbook of Robotics*. Springer, pp. 921–941.

Peng, X., Liu, K., and Jin, Y. (2016). "A dynamic optimization approach to the design of cooperative co-evolutionary algorithms". In: *Knowledge-Based Systems* 109, pp. 174 –186.

Pitla, S. K. (2012). "Development of Control Architectures for Multi-robot Agricultural Field Production Systems". PhD thesis.

Popovici, E. and De Jong, K. (2005). "A dynamical systems analysis of collaboration methods in cooperative co-evolution". In: *AAAI Fall Symposium on Coevolutionary and Coadaptive Systems*. AAAI Press.

Popovici, E. and De Jong, K. (2006). "The dynamics of the best individuals in co-evolution". In: *Natural Computing* 5 (3), pp. 229–255.

Popovici, E., Bucci, A., Wiegand, R. P., and De Jong, E. D. (2012). "Coevolutionary principles". In: *Handbook of Natural Computing*. Springer, pp. 987–1033.

Potter, M. A. and De Jong, K. A. (1994). "A cooperative coevolutionary approach to function optimization". In: *Parallel Problem Solving from Nature (PPSN)*. Springer, pp. 249–257.

Potter, M. A. and De Jong, K. A. (2000). "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents". In: *Evolutionary Computation* 8 (1), pp. 1–29.

Potter, M. A., Meeden, L. A., and Schultz, A. C. (2001). "Heterogeneity in the Coevolved Behaviors of Mobile Robots: The Emergence of Specialists". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, pp. 1337–1343.

Pugh, J. K., Soros, L. B., and Stanley, K. O. (2016). "Quality Diversity: A New Frontier for Evolutionary Computation". In: *Frontiers in Robotics and AI* 3, pp. 1–40.

Rahmattalabi, A., Chung, J. J., Colby, M., and Tumer, K. (2016). "D++: Structural credit assignment in tightly coupled multiagent domains". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, pp. 4424–4429.

Ramani, R. G., Subramanian, R, and Sindurathy, M (2008). "Strategies of teams in soccerbots". In: *International Journal of Advanced Robotic Systems* 5 (4), pp. 351–360.

Rawal, A., Rajagopalan, P., and Miikkulainen, R. (2010). "Constructing competitive and cooperative agent behavior using coevolution". In: *IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE Press, pp. 107–114.

Risi, S., Hughes, C. E., and Stanley, K. O. (2010). "Evolving plastic neural networks with novelty search". In: *Adaptive Behavior* 18 (6), pp. 470–491.

Rousseeuw, P. J. (1987). "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65.

Rubenstein, M., Ahler, C., and Nagpal, R. (2012). "Kilobot: A low cost scalable robot system for collective behaviors". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, pp. 3293–3298.

Rudolph, G. (1994). "Convergence analysis of canonical genetic algorithms". In: *IEEE Transactions on Neural Networks* 5 (1), pp. 96–101.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A modern approach*. Prentice Hall.

Sammon, J. W. (1969). "A nonlinear mapping for data structure analysis". In: *IEEE Transactions on Computers* 18 (5), pp. 401–409.

Scheepers, C. and Engelbrecht, A. P. (2014). "Competitive coevolutionary training of simple soccer agents from zero knowledge". In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press, pp. 1210–1217.

Schmidt, M. and Lipson, H. (2011). "Age-Fitness Pareto Optimization". In: *Genetic Programming Theory and Practice VIII*. Vol. 8. Genetic and Evolutionary Computation. Springer, pp. 129–146.

Sheh, R., Schwertfeger, S., and Visser, A. (2016). "16 Years of RoboCup Rescue". In: *KI - Künstliche Intelligenz* 30 (3), pp. 267–277.

Silva, F., Correia, L., and Christensen, A. L. (2016a). "Evolutionary Robotics". In: *Scholarpedia* 11 (7), p. 33333.

Silva, F., Duarte, M., Correia, L., Oliveira, S. M., and Christensen, A. L. (2016b). "Open Issues in Evolutionary Robotics". In: *Evolutionary Computation* 24 (2), pp. 205–236.

Simmons, R., Singh, S., Hershberger, D., Ramos, J., and Smith, T. (2001). "First results in the coordination of heterogeneous robots for large-scale assembly". In: *Experimental Robotics VII*. Springer, pp. 323–332.

Simpson, C. (2011). "The evolutionary history of division of labour". In: *Proceedings of the Royal Society of London B*, rspb20110766.

Smith, A. (1776). *An Inquiry into the Nature and Causes of the Wealth of Nations*. W. Strahan and T. Cadell.

Soria, N. F., Colby, M. K., Tumer, I. Y., Hoyle, C., and Tumer, K. (2016). "Design of Complex Engineering Systems Using Multiagent Coordination". In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE)*. ASME, V02AT03A001.

Stanley, K. and Miikkulainen, R. (2002). "Evolving neural networks through augmenting topologies". In: *Evolutionary Computation* 10 (2), pp. 99–127.

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). "A hypercube-based encoding for evolving large-scale neural networks". In: *Artificial life* 15 (2), pp. 185–212.

Stone, P. and Veloso, M. (2000). "Multiagent systems: A survey from a machine learning perspective". In: *Autonomous Robots* 8 (3), pp. 345–383.

Stone, P., Kuhlmann, G., Taylor, M. E., and Liu, Y. (2005). "Keepaway Soccer: From Machine Learning Testbed to Benchmark". In: *Robot Soccer World Cup IX*. Vol. 4020. LNCS. Springer, pp. 93–105.

Sukhatme, G, Montgomery, J. F., and Vaughan, R. T. (2002). "Experiments with cooperative aerial-ground robots". In: *Robot Teams: From Diversity to Polymorphism*. A K Peters, pp. 345–368.

Suzuki, Y. and Arita, T. (2006). "A comprehensive evaluation of the methods for evolving a cooperative team". In: *Artificial Life and Robotics* 10 (2), pp. 157–161.

Trianni, V. (2008). *Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots*. Vol. 108. Studies in Computational Intelligence. Springer.

Trianni, V. and López-Ibáñez, M. (2015). "Advantages of Task-Specific Multi-Objective Optimisation in Evolutionary Robotics". In: *PLoS ONE* 10 (8), e0136406.

Trianni, V., Nolfi, S., and Dorigo, M. (2008). "Evolution, self-organization and swarm robotics". In: *Swarm Intelligence*. Natural Computing Series. Springer, pp. 163–191.

Trueba, P., Prieto, A., Caamaño, P., Bellas, F., and Duro, R. J. (2011). "Task-driven species in evolutionary robotic teams". In: *Foundations on Natural and Artificial Computation: 4th International Work-Conference on the Interplay Between Natural and Artificial Computation*. Vol. 6686. LNCS. Springer, pp. 138–147.

Uchibe, E. and Asada, M. (2006). "Incremental coevolution with competitive and cooperative tasks in a multirobot environment". In: *Proceedings of the IEEE* 94 (7), pp. 1412–1424.

Uchibe, E., Nakamura, M., and Asada, M. (1998). "Co-evolution for cooperative behavior acquisition in a multiple mobile robot environment". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE Press, pp. 425–430.

Uchibe, E., Yanase, M., and Asada, M. (2002). "Behavior generation for a mobile robot based on the adaptive fitness function". In: *Robotics and Autonomous Systems* 40 (2-3), pp. 69–77.

Waibel, M., Keller, L., and Floreano, D. (2009). "Genetic team composition and level of selection in the evolution of cooperation". In: *IEEE Transactions on Evolutionary Computation* 13 (3), pp. 648–660.

Watson, R. A., Ficiei, S., and Pollack, J. B. (1999). "Embodied evolution: Embodying an evolutionary algorithm in a population of robots". In: *IEEE Congress on Evolutionary Computation (CEC)*. Vol. 1. IEEE Press.

Werfel, J., Petersen, K., and Nagpal, R. (2014). "Designing Collective Behavior in a Termite-Inspired Robot Construction Team". In: *Science* 343 (6172), pp. 754–758.

White, D. R. (2012). "Software review: the ECJ toolkit". In: *Genetic Programming and Evolvable Machines* 13 (1), pp. 65–67.

Whitley, L. D. (1991). "Fundamental Principles of Deception in Genetic Search". In: *Foundations of Genetic Algorithms (FOGA)*. Morgan Kaufmann, pp. 221–241.

Wickham, H. (2016). *ggplot2: elegant graphics for data analysis*. Springer.

Wiegand, R. P. (2003). "An Analysis of Cooperative Coevolutionary Algorithms". PhD thesis. George Mason University.

Wiegand, R. P. and Potter, M. A. (2006). "Robustness in cooperative coevolution". In: *Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, pp. 369–376.

Wiegand, R. P., Liles, W. C., and De Jong, K. A. (2001). "An empirical analysis of collaboration methods in cooperative coevolutionary algorithms". In: *Genetic and Evolutionary Computation Conference (GECCO)*. Morgan Kaufmann, pp. 1235–1245.

Wiegand, R. P., Liles, W. C., and De Jong, K. A. (2002). "Analyzing cooperative coevolution with evolutionary game theory". In: *IEEE Congress on Evolutionary Computation (CEC)*. Vol. 2. IEEE Press, pp. 1600–1605.

Wilson, E. O. and Hölldobler, B. (2005). "Eusociality: origin and consequences". In: *Proceedings of the National Academy of Sciences* 102 (38), pp. 13367–13371.

Yang, J., Liu, Y., Wu, Z., and Yao, M. (2012). "The evolution of cooperative behaviours in physically heterogeneous multi-robot systems". In: *International Journal of Advanced Robotic Systems* 9 (253), pp. 1–10.

Yitzhaki, S. et al. (2003). "Gini's mean difference: A superior measure of variability for non-normal distributions". In: *Metron* 61 (2), pp. 285–316.

Yliniemi, L. and Tumer, K. (2016). "Multi-objective multiagent credit assignment in reinforcement learning and NSGA-II". In: *Soft Computing* 20 (10), pp. 3869–3887.

Yong, C. H. and Miikkulainen, R. (2009). "Coevolution of Role-Based Cooperation in Multiagent Systems". In: *IEEE Transactions on Autonomous Mental Development* 1 (3), pp. 170–186.

Appendices

Appendix A

Experimental Details

A.1 Common Parameters

NEAT The implementation of the NEAT neuroevolution algorithm (Stanley and Miikkulainen, 2002) is based on the NEAT4J open-source software (`http://neat4j.sourceforge.net/`). In all the experiments in this thesis that used the NEAT algorithm, we use the parameters listed in Table A.1, unless indicated otherwise.

TABLE A.1: Default parameters of NEAT. These parameters were used unless explicitly indicated otherwise.

Parameter	Value	Parameter	Value
Population size	150	Target species count	5
Crossover probability	20%	Recurrency allowed	yes
Mutation prob.	25%	Add link prob.	5%
Add node prob.	3%	Mutate bias prob.	30%
Toggle link prob.	0%	Weight replaced prob.	0%
Excess coefficient	1	Disjoint coefficient	1
Weight coefficient	0.4	Survival threshold	20%
Compatibility change	0.05	Max. weight mutation	0.5
Max. bias mutation	0.1	Extinction Life Events	no
Species old threshold	80	Species youth threshold	10
Species old penalty	70%	Species youth boost	120%

Novelty Search The algorithm for computing novelty scores is implemented as described in Sections 2.5.2 and 2.5.3: the k-nearest neighbours among the current population and the archive are used for the computation of novelty score; the archive is composed of randomly chosen individuals, with each evolved individual having a fixed probability of being added to the archive; and the archive size is bounded, with randomly chosen individuals removed when necessary to allow space for new ones. Table A.2 lists the default parameters for the novelty search implementation.

TABLE A.2: Default parameters of novelty search. These parameters were used unless explicitly indicated otherwise.

Parameter	Value	Parameter	Value
Novelty k-nearest	15	Add archive criteria	random
Add archive probability	2.5%	Maximum archive size	2000

NEAT + NSGA-II NEAT is incompatible with most popular multiobjective evolutionary algorithms, and NSGA-II (Deb et al., 2002) in particular, as it requires a single fitness score to select and speciate the individuals in the population. Lehman and Stanley, (2011b) use an approach for making NEAT compatible with NSGA-II, in which the speciation is completely removed from NEAT. Having a single species thus allows for the elitism selection of NSGA-II to be directly employed in NEAT. The speciation mechanism of NEAT, however, plays an important part in the algorithm, as it protects topological innovations and maintains genetic diversity in the population (Stanley and Miikkulainen, 2002). We adopt a different approach, in which we obtain a single score from the multiple objectives, a common practice known as scalarization (Hwang and Masud, 2012). We use the NSGA-II algorithm to sort the individuals according to their objective scores, and then assign them a single score (that respects that order), which is used by the NEAT algorithm. We therefore lose the elitist selection of NSGA-II, but preserve its Pareto-based ranking mechanism and the NEAT algorithm in its entirety.

The Pareto fronts and crowding distance for all individuals in the population are first calculated according to NSGA-II (Deb et al., 2002), using the `fast-non-dominated-sort` and the `crowding-distance-assignment` algorithms. We then obtain the selection score for each individual in the population according to:

$$i_{\text{score}} = \max_{\text{rank}} - i_{\text{rank}} + \begin{cases} 1 & i_{\text{distance}} = \infty \\ \dfrac{i_{\text{distance}}}{N} & \text{otherwise} \end{cases} \tag{A.1}$$

where i_{rank} is the non-domination rank (lower is better), i_{distance} is the crowding distance (higher is better), $\max_{\text{rank}}$ is the highest rank in the current population, and N is the number of objectives (2 in the case of novelty and fitness combination). This formula ensures the same ranking of individuals as the original NSGA-II algorithm.

A.2 Predator-prey Task

The parameters of the experiments with the simulated predator-prey task (Section 3.4) are listed in Table A.3.

TABLE A.3: Parameters used in the experiments with the predator-prey task. The time (s – step) and space units (u – unit) are abstract.

Parameter	Value	Parameter	Value
Predator-prey task			
Arena size	100×100 u	Prey placement area	10×10 u
Maximum trial length	300 s	Number of predators	2–7
Predator linear speed	1 u/s	Predator turn speed	45°/s
Prey speed	1 u/s	Prey escape distance (V)	4–13 u
Genetic algorithm			
Population size	150	Elite size	5
Tournament size	5	Mutation type	Gaussian
Gene mutation probability	5%	Mutation σ	0.5
Crossover probability	50%	Crossover type	one point
Generations	500		

Neural network architecture of the predator agents Each predator is controlled by a fixed-topology Jordan network (Jordan, 1997), a simple recurrent network with a state layer connected to the output neurons. The network has two inputs, eight hidden neurons, two outputs, and each layer is fully connected to the layer that it feeds. The structure of the neural network and the number of hidden neurons were tuned empirically in preliminary experiments with the predator-prey task. Feed-forward networks and Elman networks (Elman, 1990) were also tested, and the number of hidden neurons was varied from three to ten. We chose the architecture that yielded the highest fitness scores in the preliminary experiments: a Jordan network with eight hidden neurons.

Prey movement The prey moves away from the nearby predators, taking into account their distance. Let $\mathbf{y}_t$ be the current position of the prey, $\mathcal{R}_t$ the set composed of the current positions of the predators, V the prey escape distance, and S the prey escape speed. The position of the prey is updated every control step according to:

$$\begin{aligned} \mathcal{R}'_t &= \{\mathbf{r} \in \mathcal{R}_t : ||\mathbf{y}_t - \mathbf{r}|| < V\} \\ \mathbf{v} &= \langle 0, 0 \rangle + \sum_{\mathbf{r} \in \mathcal{R}'_t} \frac{\mathbf{y}_t - \mathbf{r}}{||\mathbf{y}_t - \mathbf{r}||^2} \\ \mathbf{y}_{t+1} &= \mathbf{y}_t + S \cdot \hat{\mathbf{v}} \end{aligned} \tag{A.2}$$

A.3 Cooperative Foraging Task

Table A.4 lists the parameters of the cooperative foraging task (Section 3.5.1).

TABLE A.4: Parameters for the cooperative foraging task. The time (s – step) and space units (u – unit) are abstract.

Parameter	Value	Parameter	Value
Arena size	150×150 u	Maximum trial length	1000 s
Max. linear speed	1 u/s	Max. rotation speed	23°/s
Sensor range	25 u	Min. actuator activation time	25 s
Item diameter	12 u	Agent diameter	4 u
NEAT Generations	500		

Item actuator An agent's item actuator becomes active if the respective output exceeds the threshold of 0.5 (the outputs' range is $[0, 1]$). In case both outputs exceed the threshold, the one with the highest value is activated.

A.4 Herding Task

Table A.5 lists the configuration of the herding task (Section 3.5.2).

Sheep movement The sheep moves away from the closest shepherd, if it is closer than the action range A. Let $\mathbf{s}_t$ be the current position of the sheep, $\mathbf{h}_t$ the current position of the closest shepherd, V_S the sheep speed, and A the action range (see

TABLE A.5: Parameters for the herding task. The time (s – step) and space units (u – unit) are abstract.

Parameter	Value	Parameter	Value
Arena size	150×150 u	Max. trial length	500 s
Sheep speed (V_S)	1 u/s	Fox speed (V_F)	1 u/s
Shepherd linear speed	1 u/s	Shepherd turn speed	23°/s
Action range (A)	5 u	Shepherd sensor range	25 u
NEAT Generations	500		

Table A.5). The position of the sheep is updated every control step according to:

$$\mathbf{s}_{t+1} = \begin{cases} V_S \cdot \dfrac{\mathbf{s}_t - \mathbf{h}_t}{||\mathbf{s}_t - \mathbf{h}_t||} & ||\mathbf{s}_t - \mathbf{h}_t|| < A \\ \mathbf{s}_t & \text{otherwise} \end{cases} \tag{A.3}$$

Fox movement The fox moves away from the closest shepherd, or in the case there is none nearby, it moves in the direction of the estimated future position of the sheep. Let $\mathbf{s}_t$ be the position of the sheep at the instant t, $\mathbf{f}_t$ the current position of the fox, $\mathbf{h}_t$ the position of the closest shepherd, V_S the sheep speed, and V_F the fox speed (see Table A.5). The position of the fox is updated every control step according to:

$$\mathbf{f}_{t+1} = V_F \cdot \begin{cases} \dfrac{\mathbf{f}_t - \mathbf{h}_t}{||\mathbf{f}_t - \mathbf{h}_t||} & ||\mathbf{f}_t - \mathbf{h}_t|| < A \\ \dfrac{\mathbf{f}_t - \mathbf{s}'}{||\mathbf{f}_t - \mathbf{s}'||} & \text{otherwise, where :} \end{cases} \tag{A.4}$$

$$\mathbf{s}' = \mathbf{s}_t + \frac{\mathbf{s}_t - \mathbf{s}_{t-10}}{10} \cdot \min\left\{50, \frac{||\mathbf{f}_t - \mathbf{s}_t||}{V_S}\right\}$$

A.5 Aquatic Predator-prey Task

Table A.6 lists the configuration of the aquatic predator-prey task (Section 4.1), including both the setup used during evolution and the setup used for the evaluation with the real robotic system. Table A.7 lists the physical properties of the robotic platform (Section 4.2).

Prey movement The prey *tries* to moves away from the nearest predator, if it closer than the escape distance (V), otherwise it stops. Contrary to the simulated predator-prey task in Chapter 3, the prey typically can not move immediately in the opposite direction of the predator, due to the physical limitations imposed by the real robots with differential drive. The prey attempts to move in that direction as quickly as possible, rotating at full speed if necessary, but its actual movement depends on its current orientation and speed, turning speed, as well as the natural uncertainty of the movement in water.

Software The simulation of the task was conducted in the JBotAquatic simulator (`https://github.com/BioMachinesLab/drones/tree/master/JBotAquatic`), an extension of JBotEvolver (Duarte et al., 2014). The software used

TABLE A.6: Parameters used for the setup of the aquatic predator-prey task.

Parameter	Value	Parameter	Value
Task setup for the evolutionary process			
Virtual arena boundaries	75×75 m	Predator placement area	10×10 m
Prey placement distance	$[20, 35]$ m	Prey capture distance	2 m
Prey escape distance (V)	10 m	Trial time limit	75 s
Predator sensor range	40 m	NEAT generations	250
Simulation noise			
GPS location error	up to 1.8 m	Compass reading error	$[-10, 10]^\circ$
Motor response delay	500 ms	Compass offset	$[-9, 9]^\circ$
Motor speed offset	$[-10, 10]\%$	Motor output noise	$[-5, 5]\%$
Prey speed offset	$[-25, 0]\%$	Prey escape direction error	$[-5, 5]^\circ$
Task setup for the real-robot experiments			
Virtual arena boundaries	100×100 m	Trial time limit	100 s
Prey placement distance	25, 30, 35 m		
The remaining parameters are the same as the evolution version			

TABLE A.7: Measured movement dynamics and physical properties of the robotic platform that was used for both the predators and the prey. These parameters were used to model the robot in simulation. The full specification of the robotic platform is published in (Costa et al., 2016).

Parameter	Value	Parameter	Value
Dimensions (l×w×h)	65×40×15	Weight	3 Kg
Minimum linear speed	0.3 m/s	Maximum linear speed	1.7 m/s
Maximum turning rate	90°/s	Time from stop to full speed	1 s
Time from full speed to stop	5 s	Battery autonomy	2–3 h
Wi-Fi broadcast range	40 m†	Position broadcast frequency	1 Hz

† The actual range can be different in the real robot experiments, depending on interferences in the wireless communication.

to control the real robots is also available in the repository: `https://github.com/BioMachinesLab/drones`.

A.6 Aerial-ground Foraging Task

The parameters for the aerial-ground foraging task (Section 5.1) are listed in Table A.8.

Camera-based sensors The sensors for the detection of the aerial robot (by the ground robot), and for the detection of the items and the ground robot (by the aerial robot), are based on a vertical camera sensor with a field of view of 60° (α). This means that the actual range of the sensors depends on the current altitude (h) of the aerial robot. The range of these sensors is given by:

$$r = h \cdot \tan\left(\frac{\alpha}{2}\right) \tag{A.5}$$

TABLE A.8: Parameters of the aerial-ground foraging task. The NEAT algorithm and novelty search used the default parameters listed in Table A.1.

Parameter	Value	Parameter	Value
Ground robot diameter	8 cm	Aerial robot diameter	40 cm
Max. trial duration	200 s	Total arena size	5500×350 cm
Number of items	6	Item placement area (x6)	150×150 cm
Item diameter	5 cm	Ground r. linear speed	15 cm/s
Ground R. turn speed	180°/s	Aerial R. linear speed†	100 cm/s
Aerial R. linear accel†	10 cm/s^2	Aerial R. rotation speed	90°/s
Aerial R. rotation accel	15°/s^2	Aerial R. max altitude	∞
Ground R. item sensor	10 cm	Camera sensors FOV	60°
Camera sensors vert. range	250 cm	Camera sensors hor. range‡	[0,144] cm

† In any direction. ‡ Depending on the current altitude.

Choice of representative in multi-objective algorithms Since the multiobjective algorithm used in both *MOEA* and *NS* uses a Pareto-based ranking of the individuals (see Section 5.3.1), the definition of the *best* individual (used as the population representative) is ambiguous. We empirically evaluated different strategies for choosing the representative in this case:

Best: The highest-fitness (number of items collected) individual of the previous generation is chosen as the population representative.

Random: One randomly chosen individual from the current population is chosen as the representative.

Pareto: One randomly chosen individual belonging to the first (non-dominated) Pareto front is used as the representative of the population.

Max: The individuals that obtained the highest scores in each of the objectives are chosen as the representatives of the population. When the same individual obtained the highest scores for all objectives, only that individual is chosen as the representative. Multiple collaborations can thus be used to evaluate each individual. The individuals are scored according to the collaboration that achieved the highest fitness score (number of items collected).

The results are presented in Figure A.1, with 30 independent evolutionary runs performed for the *Best* strategy, for each task variant, and 15 evolutionary runs for each of the other strategies. The results show that the *Best* strategy is overall the highest performing strategy, and it was therefore used for the *MOEA* and *NS* methods. For both methods, the *Best* strategy is never significantly outperformed by the other strategies ($p < 0.05$, Mann-Whitney U test) in all task variants.

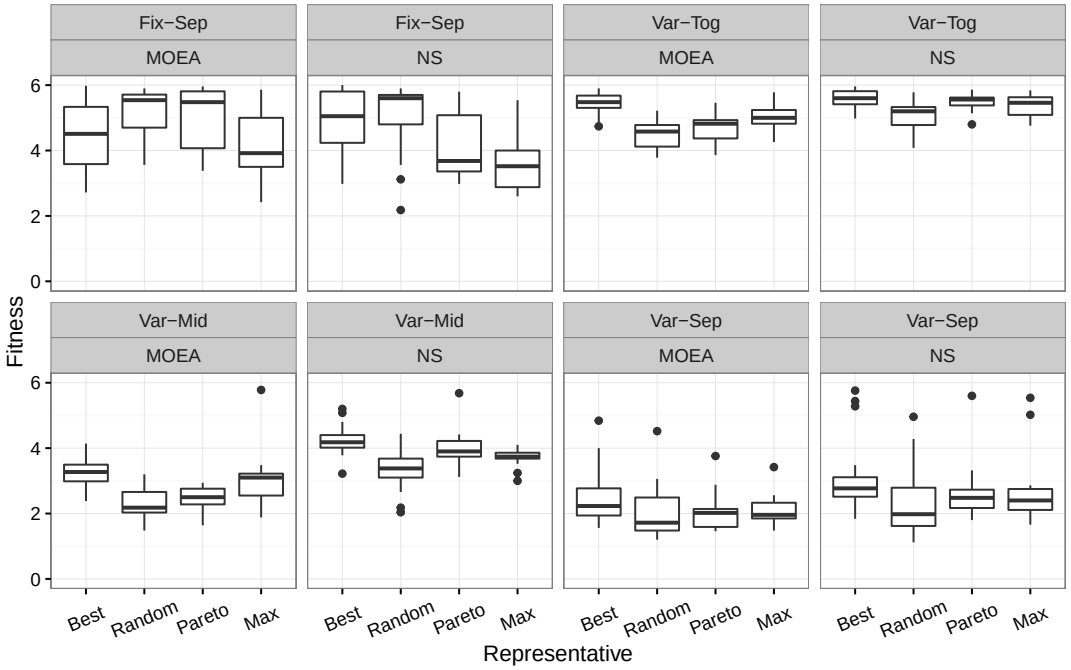


FIGURE A.1: Boxplots of the highest fitness score achieved in each evolutionary run, comparing the different representative selection strategies.

A.7 Coverage Task

The parameters used for the genetic algorithm in the coverage task (Section 6.3) are listed in Table A.9.

TABLE A.9: Evolutionary algorithm parameters used for the abstract coverage task.

Parameter	Value	Parameter	Value
Population size	100	Elite size	5
Tournament size	2	Mutation type	Gaussian
Mutation σ	0.25	Gene mutation prob.	20%
Crossover type	one point	Crossover prob.	50%
Evaluation budget	1.5M (Unless indicated otherwise)		

A.8 Multi-rover Foraging Task

The parameters used for the multi-rover foraging task (Section 6.4.2) are listed in Table A.10.

TABLE A.10: Multi-rover foraging task parameters.

Task environment	
Number of robots	10
Total number of items	30
Environment 1 items	15×$\{R_1, R_2\}$
Environment 2 items	6×$\{R_1, R_2, R_3, R_4, R_5\}$
Arena size (bounded by walls)	1000 × 1000 cm
Robot starting area (in the centre)	125 × 125 cm
Robot diameter	10 cm
Max. simulation time	100 s
# of independent simulations per evaluation	5
Minimum sensor resolution activation time	10 s
Item capture distance	5 cm
Robot actuators (transformed from $[0, 1]$)	
m_ρ: Linear speed	[0,12.5] cm/s
m_ϕ: Turning speed	[0,112.5]°/s
R: Sensor resolution	[1,$N^\dagger$]
Robot sensors (normalised to $[-1, 1]$)	
w_l, w_r: 2 binary whisker sensors that indicate the presence of the walls	±30°, 12.5 cm range
$k_{1..6}$: 6 evenly distributed circular sectors, returning the distance to the closest item	Unlimited range
$r_{1..4}$: 4 circular sectors returning the distance to the closest agent	250 cm range
n_R: Closest agent's sensor resolution	[1,$N^\dagger$]
† N is the number of different item types.	
NEAT parameters	
Population size	100
Evaluation budget	5M
Remaining parameters are the default in Table A.1	

A.9 Soccer Task

The parameters used for the soccer task (Section 6.4.3) are listed in Table A.11. We additionally provide a brief description of the manually programmed soccer controller below.

Manually programmed soccer strategy The manually programmed soccer agent controller was based in the *AIKHomoG* control strategy (`http://www.cs.cmu.edu/~trb/TeamBots/Domains/SoccerBots/`. A simplified description of the manually programmed strategy is provided in Algorithm 10 (a number of strategy details have been omitted for brevity). The full strategy implementation can be found at `https://github.com/jorgemcgomes/mase/blob/master/src/mase/app/soccer/AIKAgent.java`.

The `freeKickDirection`(`x`) and `freeMoveDirection`(`x`) functions return the closest direction to `x` for kicking/moving without any impeding obstacles (walls or other agents). The `positioningForce`() is a weighted sum of several force vectors, which have the following effects:

TABLE A.11: Soccer task parameters.

Task environment	
Number of robots per team	5
Field size	274×152 cm
Robot diameter	8 cm
Ball diameter	4 cm
Ball slip deceleration	3.5 cm/s^2
Ball roll deceleration	0.3 cm/s^2
Ball slip to roll	70% of kick speed
Ball minimum speed	2 cm/s
Ball coefficient of restitution (rebound)	0.85
Goal width	50 cm
Robots starting area	60×60 cm
Game time limit	100 s
Ball stuck: moves less than 10 cm in 10 s with an agent close to it.	
Number of games per evaluation	10
Robot actuators (transformed from $[0, 1]$)	
m_ρ: Linear speed	[0,10] cm/s
m_ϕ: Movement direction†	[-180,180] °
k_ρ: Kick speed	[10,40] cm/s
k_ϕ: Kick direction†	[-180,180] °
Robot sensors (normalised to $[-1, 1]$)	
$dt_1 \ldots dt_4$: 4 evenly distributed circular sectors, returning the distance‡ to the closest teammate in the respective sector	
$do_1 \ldots do_4$: Same as dt, but for the opponents	
b_ρ, b_ϕ: Distance‡ and relative angle† to the ball	
gt_ρ, gt_ϕ: Distance‡ and relative angle† to own goal	
go_ρ, go_ϕ: Distance‡ and relative angle† to opponents' goal	
† The robots have a fixed orientation, facing the opponents' goal line	
‡ Infinite range, normalised according to the field's diagonal D	
NEAT parameters	
Population size	100
Evaluation budget (Soccer-80%)	2,5M
Evaluation budget (Soccer-100%)	5M
Remaining parameters are the default in Table A.1	

- Keep distance from the teammates.
- Keep distance from the walls.
- Go to the goalie position if no other teammate is currently assuming that role.
- Go to the left-offensive position if no other teammate is in that position.
- Go to the right-offensive position if no other teammate is in that position.

We implemented a number of improvements in the original *AIKHomoG* control strategy in order for it to present a meaningful challenge to the evolutionary algorithms used in our experiments. In preliminary experiments, the evolutionary process quickly exploited behavioural flaws in the original *AIKHomoG* strategy. Compared to the original strategy, the following major changes were made:

Algorithm 10 Manually programmed strategy of a soccer agent.

```
1:  if agent is closest to ball then
2:      if agent has the ball then
3:          g ← direction to the opponents' goal
4:          g′ ← freeKickDirection(g)
5:          Kick the ball in the direction g′ at full power
6:      else
7:          b ← direction to the ball
8:          b′ ← freeMoveDirection(b)
9:          Move in the direction b′ at 100% speed
10: else
11:     f ← positioningForce()
12:     if ||f|| > force threshold then
13:         f′ ← freeMoveDirection(f)
14:         Move in the direction f′ at 75% speed
```

- The obstacle detection was improved – the walls are now also included in the obstacle list.
- When shooting the ball the teammates are not considered obstacles, so that it is possible to shoot the ball towards the teammates (passing).
- When calculating the shoot direction, the opponents' movement range is taken into account, in order to avoid shots that are easily intercepted by the opponents.
- The goal of the player's team is considered an obstacle in order to avoid own-goals.
- The agent goes directly towards the ball to position itself to shoot, as in our task the agents can shoot in any direction.
- The goalie behaviour was improved – the agent acting as goalie is more effective in tracking the ball, moving along the goal line.

Appendix B

Evolution and Simulation Framework

To support the experiments described in this thesis, we developed MASE (Multi-Agent Systems Evolution), a Java-base open-source framework built over existing libraries. The source code is available at `https://github.com/jorgemcgomes/mase`.

B.1 Architecture

The developed framework is based on two pre-existing systems, ECJ and MASON, described below. The integration of the software modules is illustrated in Figure B.1.

ECJ: Java-based Evolutionary Computation Research System, developed by Sean Luke et al. at George Mason University's ECLab (Evolutionary Computation Laboratory).[1] ECJ is one of the most popular evolutionary computation toolkits, aimed at all forms of evolutionary computation. Due to its well-engineered structure, which makes heavy use of Java inheritance, abstraction and pattern-oriented design, ECJ is able to support many alternative methods for common functions, such as population initialisation, selection and variation operators, without requiring additional user-written code (White, 2012). We use ECJ as the backbone for the MASE framework, leveraging the existing evolutionary algorithms, parameter configuration system, and overall organisation.

MASON: A widely used simulation library in Java, developed by Sean Luke et al. at George Mason University's ECLab and Center of Social Complexity (Luke et al., 2005).[2] MASON is a fast discrete-event multiagent simulation library, designed to be the foundation for large custom-purpose Java simulations, and also to provide functionality for many lightweight simulation needs. We extended MASON to provide basic functionality for multirobot systems simulation, and implemented our multirobot tasks over it.

Below, we describe the main features implemented in the MASE framework.

[1] `http://cs.gmu.edu/~eclab/projects/ecj/`
[2] `http://cs.gmu.edu/~eclab/projects/mason/`

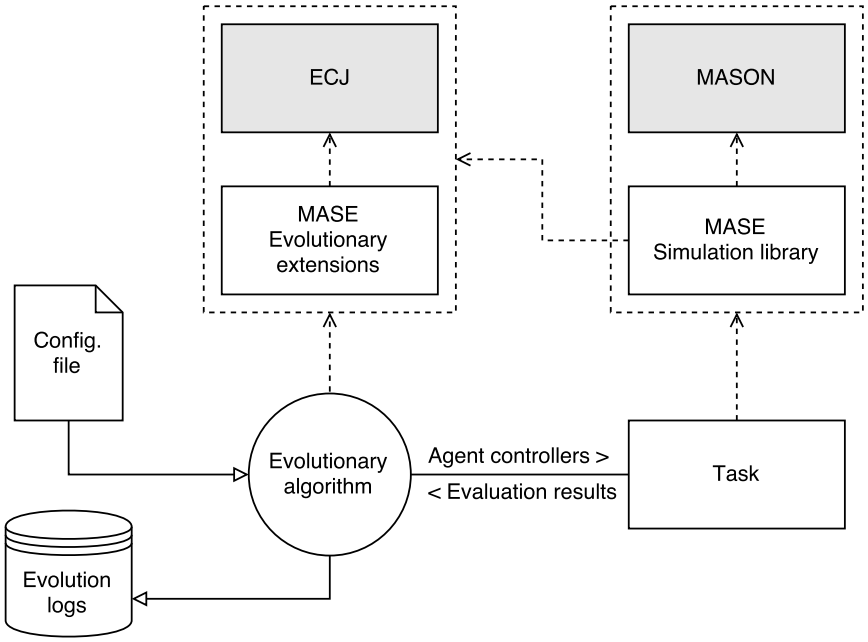


FIGURE B.1: Articulation of the software components in the MASE framework. The evolution module is independent from the simulation module. The simulation module only depends on the interfaces of the evolution module that define the agent controllers and evaluation functions and results.

B.2 MASE Evolutionary extensions

We extended the functionality of ECJ with the following features, in order to implement new algorithms and enable the evolution of robot controllers.

Agent controllers: Interfaces for black-box-style agent controllers, and for team controllers composed of multiple agent controllers (including homogeneous and heterogeneous teams). Implementation of neural-network based controllers, with multiple architectures supported. See package `mase.controllers`.

NEAT: Implementation of the NEAT algorithm in the ECJ system. Implementation based on the NEAT4J library[3]. See package `mase.neat`.

Extended evaluation: The ECJ system only supported traditional evaluation based exclusively on fitness scores. We extended it to support the evaluation of behaviours, including the use of multiple evaluation functions for each individual. See package `mase.evaluation`, especially the class `ExpandedFitness` and the interfaces `EvaluationFunction` and `EvaluationResult`.

Post-evaluators: Novelty-based algorithms require the individual scores to be computed after the entire population is evaluated. To enable this, we implemented *post-evaluators*, which can modify the scores of the individuals in the current population, and are run after the regular evaluation phase is completed. See class `MetaEvaluator` and interface `PostEvaluator`.

Novelty search: We implemented the novelty search algorithm supporting a multitude of implementation options, and also several methods for combining novelty scores with fitness scores. See packages `mase.novelty` and `mase.mo`.

[3] `http://neat4j.sourceforge.net/`

Hyb-CCEA: The Hyb-CCEA algorithm as described in Chapter 6. See package `mase.spec`.

Extended statistics: We implemented new *statistics* to extend the data gathered during the evolutionary process, and to enable the re-evaluation of solutions after the evolutionary process is finished. See package `mase.stat`.

Parallelisation: Due to the high computational costs of evolution and simulation, we implemented mechanisms for distributing the workload to the available computational resources, including: (i) distribution of evolutionary runs to multiple computers over the network (see `MaseManager`); (ii) submission of jobs to a HPC cluster based on the Oracle Grid Engine (see `HPCDispatcher`); and (iii) submission of jobs to the distributed computing system Conillon[4] (see package `conillon`). We additionally developed a user interface for creating and managing batches of evolution jobs, see `MaseManagerTerminal`.

B.3 MASE Simulation library

We extended the MASON simulator with basic functionality for multirobot tasks, in order to facilitate the implementation of different tasks.

Simulation problem: Interface to the evolutionary algorithm, providing the methods that can be invoked to perform the evaluation of a candidate solution. See `MasonSimulationProblem`.

Basic simulation: Extends the basic MASON simulation to run the provided evaluation functions. See `MasonSimState` and `MasonEvaluation`.

Basic agent: Embodied and situated agent that can be configured with sensors and actuators, and can be controlled by an evolved agent controller. See `SmartAgent`.

Environment objects: Entities that can be added to the environment and sensed by the agents. See `CircularObject`, `StaticPolygonObject`, `EmboddiedAgent`.

Sensors and actuators: A set of different sensors and actuators commonly used in evolutionary robotics studies (cone-type sensor, ray-based sensor, range and bearing, differential drive, etc.). See interfaces `Sensor` and `Effector`.

Generic characterisations: Implementation of generic and systematically-derived behaviour characterisations, as described in (Gomes and Christensen, 2013; Gomes et al., 2014e). See package `mase.mason.generic`.

B.4 Implemented Tasks

The following tasks are currently implemented in the MASE frnetamework (see package `mase.app`):

Pedator-prey: In this task, a group of predators (under evolution) cooperate to catch a prey that tries to escape from the nearby predators. The task is used in the experiments in Chapters 3 and 4.

[4] https://github.com/BioMachinesLab/conillon

Competitive predator-prey: In the competitive version of the predator-prey task, a single predator evolves to catch a single prey, which also evolves to escape the predator. Intended for competitive coevolution studies, not used in any published study.

Multi-rover foraging: Task where multiple agents have to cooperate to find and capture items spread throughout a closed arena. Different versions of this task have been used in Chapters 3 and 6.

Aeria-ground foraging: Foraging task where an aerial and a ground robot have to cooperate to find and capture items in the environment. Task used in Chapter 5.

Soccer: Full soccer task where a team of agents under evolution play against a team with a manually programmed strategy. Task used in Chapter 6.

Keepaway soccer: Simplified soccer task where multiple keepers (under evolution) must pass the ball within a restricted area, keeping it away from a taker (preprogrammed) that actively tries to catch it. Task used in (Gomes et al., 2014a).

Competitive keepaway soccer: Version of the keepaway soccer task where both the keepers (homogeneous in this version) and the taker are under evolution, intended to be used in competitive coevolution studies. Not used in any published study.

Herding: Task where a team of *shepherds* (under evolution) has to push *sheep* into the corral, while keeping *foxes* away from the sheep. This task was used in Chapter 3 and in (Gomes et al., 2015b).

Coverage: Abstract function optimisation task used in Chapter 6.

Go: Game of Go with neural-based agent controllers, used for experiments with competitive coevolution. Not used in any published study.

Aggregation: Classical swarm robotics tasks, where the robots start randomly spread and have to aggregate in any point of the arena. Not used in any published study.

Gate escape: In this task, a group of robots must escape through a narrow gate that closes shortly after the first robot has passed. The task was used in (Gomes et al., 2014e).

Resource sharing: In the resource sharing task, a group of robots must coordinate in order to allow each member periodical access to a single battery charging station. Task used in (Gomes et al., 2014e).

Maze navigation: Task where a single agent has to navigate through a maze in order to reach the end-point. Task used in (Gomes et al., 2015c).

B.5 Data Analysis

The analysis of the logs produced by the evolutionary algorithms is conducted with R scripts. We developed a set of general functions for loading, parsing, and analysing such data, available at `https://github.com/jorgemcgomes/mase/blob/master/R/mase.r`. The plots are produced with the *ggplot2* plotting system (Wickham, 2016).